SIMULATION OF A DIESEL-HYDRAULIC SERIES HYBRID

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Mechanical Engineering

by

Drew Anthony Lozano

March 2010

COMMITTEE MEMBERSHIP

TITLE:	SIMULATION OF A DIESEL-HYDRAULIC SERIES HYBRID
AUTHOR:	Drew Anthony Lozano
DATE SUBMITTED:	March 2010
COMMITTEE CHAIR:	Dr. James Widmann
COMMITTEE MEMBER:	Dr. John Ridgely
COMMITTEE MEMBER:	Dr. Xi Wu

ABSTRACT

SIMULATION OF A DIESEL-HYDRAULIC SERIES HYBRID

Drew Anthony Lozano

The development of hybrid vehicles is one of the current approaches to improve fuel efficiency. California Polytechnic State University San Luis Obispo's current platform is a diesel-hydraulic series hybrid. This thesis provides a MATLAB based model and simulation for this platform to be used as a development tool. The model is highly configurable and the code employs parallel processing to allow for rapid simulation of a range of inputs. A best-efficiency control strategy is implemented to select between operational modes containing optimized engine and hydraulic system parameters during the simulation. Results from the simulations are comparable to results from similar simulations.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

Chapter 1: **Introduction**

1.1 Background

Cal Poly's Hybrid Vehicle Development Team (HVDT) began as the Future Truck team. Future Truck was a competition sponsored by the Department of Energy and Ford Motor Company where students were tasked with developing a low-emission, high fuel economy Ford Explorer. The team successfully built a diesel-electric series hybrid for this competition. With the end of the Future Truck competition, the team renamed itself HVDT and began developing a diesel-hydraulic series hybrid. This project, a part of that effort, aims to design a viable hydraulic system and create a functional simulation of HVDT's concept vehicle that allows for rapid iteration with different control strategies, component properties, and vehicle parameters. The ability to quickly run simulations with different parameters would enhance the value of the program as a design tool by allowing a larger set of components to be selected from and later as a development tool when trying to optimize control strategies while adhering to time constraints.

1.2 Hybrid Vehicles

One of the goals of a hybrid vehicle is to improve fuel efficiency. This is accomplished through operating the vehicle's engine nearer to its optimum efficiency and reclaiming energy that would otherwise be turned into heat while decelerating. Decoupling the engine's output from the vehicle's immediate power requirements allow its operating points through a driving cycle to be altered. Hybrid vehicles can generally be classified based on their implementation of these aims with two parameters: powertrain configuration and method of energy storage. Common powertrains include parallel (Figure 1.2), series (Figure 1.3), parallel-series (Figure 1.4), and launch-assist configurations (Figure 1.1). Green arrows in the figures describing the powertrain configurations show the possible paths through which energy flows between components. Electric, hydraulic, and flywheel systems are used to store energy.

A vehicle with launch-assist has an energy conversion unit (ECU) connected to the drive axle which can recover kinetic energy during deceleration. The recovered energy is stored in an energy storage unit (ESU). The recovered energy in the ESU is passed through the ECU when accelerating from a stop. A

launch-assisted vehicle normally drives under power from its engine which has a direct link to the drive axle and does not have the capability of transferring energy directly from the engine to the ECU.

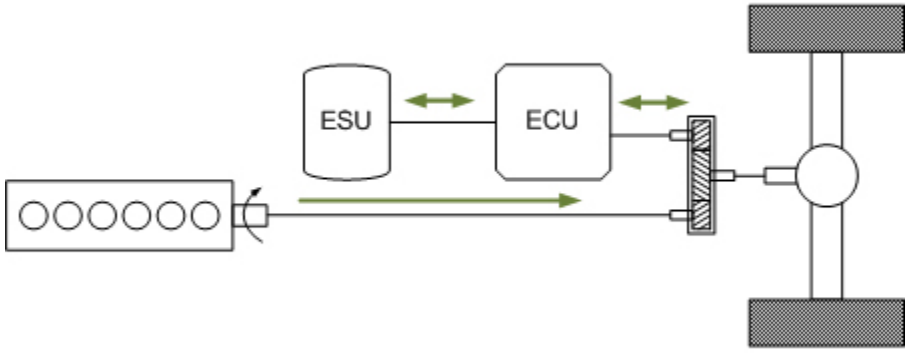


Figure 1.1: Launch-assist powertrain configuration

Parallel hybrids are similar to launch-assisted vehicles in that they have an ECU that recovers kinetic energy and a direct connection between the engine and drive axle. Unlike the launch-assisted vehicle, the parallel hybrid can charge the ESU directly through the ECU. This type of hybrid allows the vehicle to be driven by the engine or ECU individually or by a combination of the two.

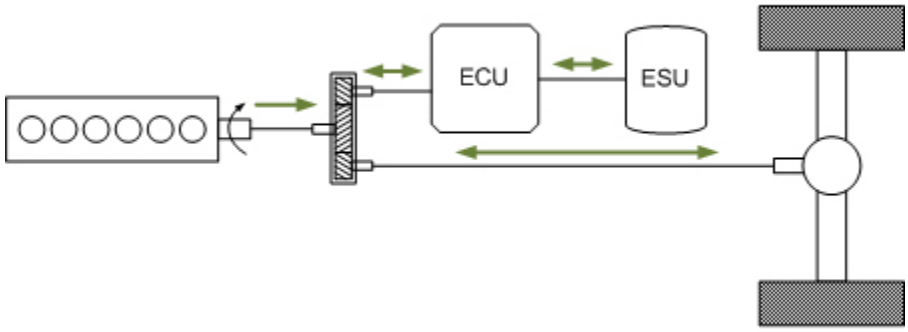


Figure 1.2: Parallel hybrid powertrain configuration

A series hybrid does not have a direct connection between the engine and drive axle. Instead, energy from the engine must pass through a pair of ECUs before reaching the drive axle. Energy passed through the

first ECU can be stored in the ESU, passed through the second ECU, or split between the ESU and second ECU. This configuration recovers kinetic energy through the second ECU.

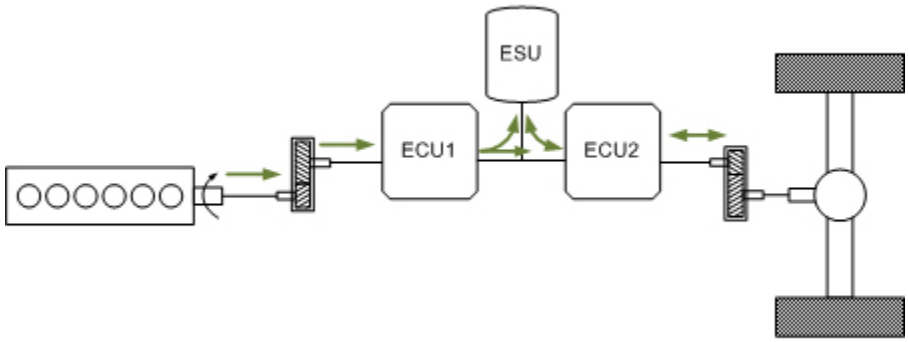


Figure 1.3: Series hybrid powertrain configuration

Parallel-series hybrids are a combination of the parallel and series configurations. The engine in this configuration is connected to the first ECU and the drive axle. The second ECU is connected to the drive axle and shares an ESU with the first ECU. A parallel-series hybrid allows the vehicle to be driven directly by the engine or by the engine and second ECU. Energy from the engine can also be split between the drive axle and the first ECU. Kinetic energy recovery is performed through the second ECU.

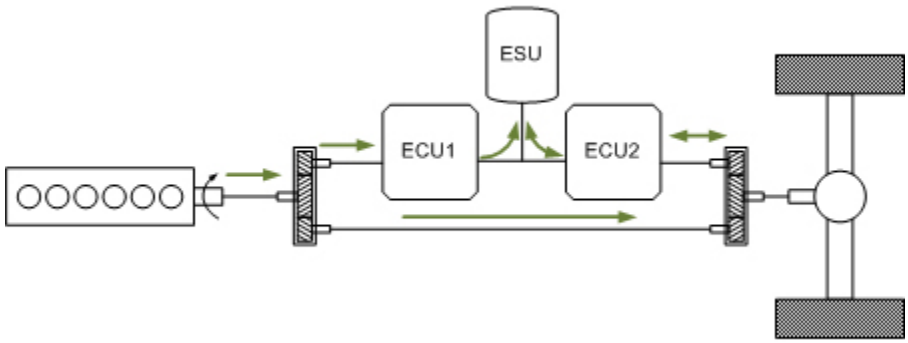


Figure 1.4: Parallel-series hybrid powertrain configuration

1.3 Simulations

An important part of the development process is creating a simulation of the vehicle such that performance of its systems can be modeled. During the design phase, simulation of a range of components aids the selection by determining their common operating regimes throughout a driving cycle. This also allows for evaluation of various control strategies before implementing them on the vehicle.

The Environmental Protection Agency (EPA) created the Stored Hydraulic Energy Research Platform Analyzer (SHERPA) as part of their research of hydraulic hybrid vehicles (Alson, et al., 2004). SHERPA is a Simulink based simulation that is a forward model where the input is a torque to achieve the desired vehicle response. They used SHERPA to investigate launch-assist and series hybrid configurations of a SUV and a midsize car on the Federal Test Procedure (FTP) city and highway driving cycles. The EPA predicted fuel economy increases of 18% (gasoline engine, engine on strategy) to 140% (variable displacement diesel engine, reduced road load, engine off strategy) for a series hybrid SUV when compared to the baseline SUV.

Buchwald et al. created a FORTRAN based simulation when exploring ways to improve the efficiency of city buses. Like the EPA simulation, it is a forward simulation where the driver commands a torque to achieve the desired vehicle response. Using a data logged driving cycle from a conventional city bus, Buchwald et al. showed a 25-30% increase in fuel efficiency for a city bus with a parallel hybrid configuration.

Chapter 2: **Hydraulic System**

2.1 Schematic

The complete hydraulic schematic for HVDT's diesel-hydraulic series hybrid is shown in Figure 2.2 which includes the pump, motors, energy storage system, flow control systems, fluid conditioning system, and safety systems. Table 2.1 contains a description of the linetypes found in the hydraulic schematic. This particular series configuration has three ECUs, one hydraulic pump and two hydraulic motors, because the vehicle retains its four wheel drive capabilities where the previous figures showed two wheel drive configurations. Figure 2.1 shows a simplified layout of HVDT's vehicle.

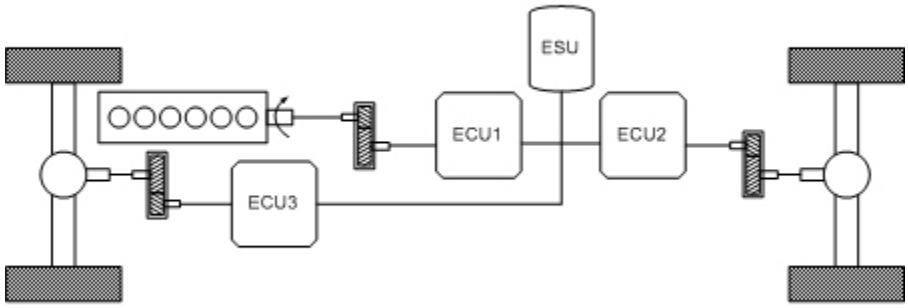


Figure 2.1: Simplified system configuration

Table 2.1: Hydraulic schematic legend

Linetype	Description
	Full Pressure Service Line
	Control Valve Service Line
	100bar Pilot
	Manifold
	Kidney Loop
	Return Line

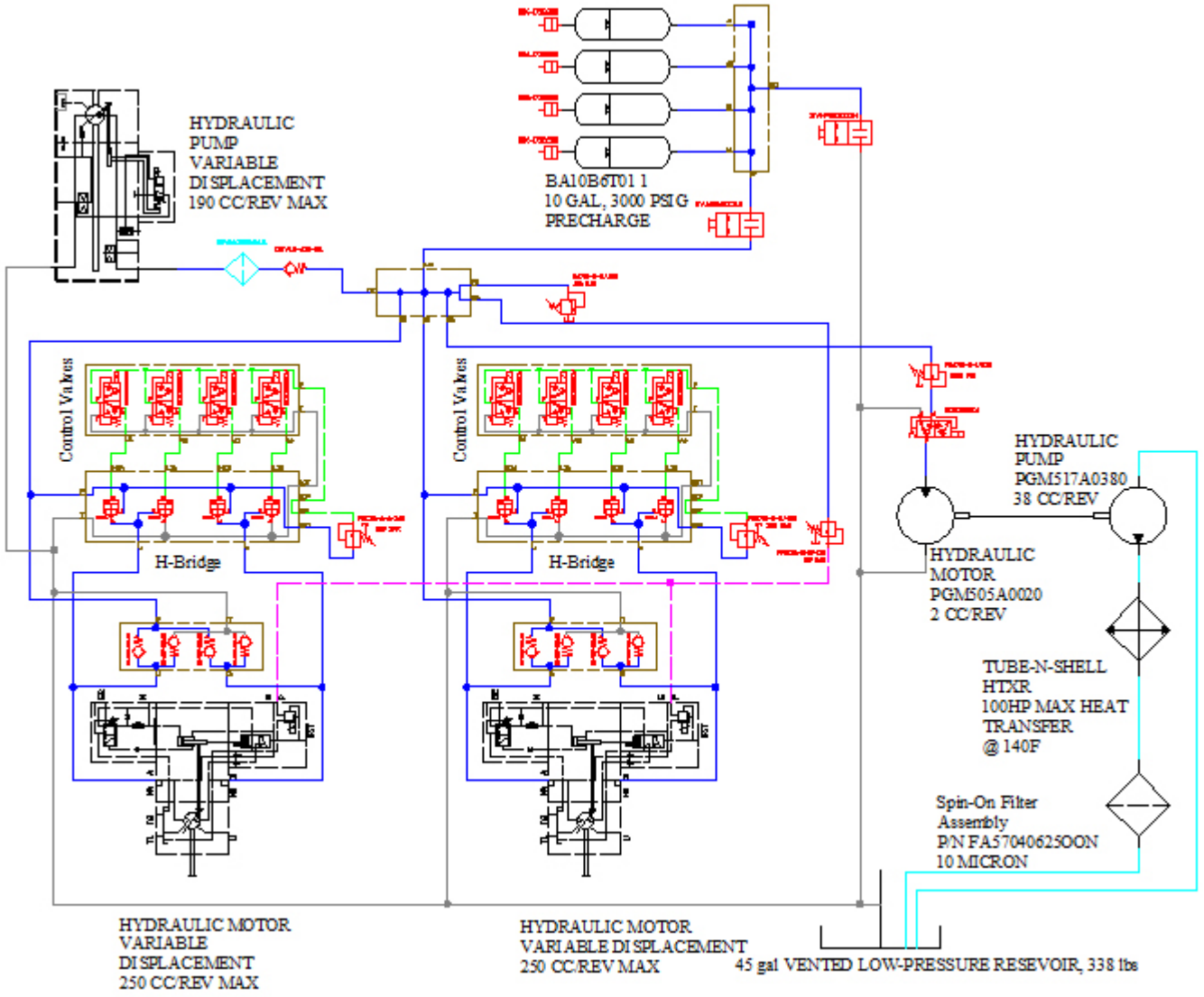


Figure 2.2: Hybrid Vehicle Development Team hydraulic schematic

2.2 Pump

The pump is a 190 cc/rev variable displacement axial piston pump which is driven by an engine through a gearbox. It was sized to absorb the power from the engine at about 1/4 of the engine speed. When driven, the pump draws fluid from the low pressure reservoir through its suction port and forces it into the high pressure side of the system. The unit is equipped with a charge pump and does not require external pilot pressure lines or a pressurized low pressure reservoir. Pilot pressure is provided internally by the charge pump. Variation of the shaft speed and displacement allows the power input to the hydraulic system to be controlled. Displacement is controlled via a solenoid where the displacement is proportional to the current flowing through the solenoid. A check valve is placed on the output of the pump to increase efficiency by preventing backflow.

2.3 Motors

Variable displacement bent axis motors are attached to the front and rear differentials. For simplicity and budgetary reasons, HVDT elected to connect the motors to the differentials without using gearboxes. The motors were sized to provide similar launch performance the original vehicle. Without a gear reduction this leads to motors with displacements much larger than otherwise necessary. The motors HVDT specified are 250 cc/rev. Both motors provide driving and braking torque for the vehicle. The pressure differential between the ports on the motor and the motor displacement provide torque at the motor's output shaft (Equation 3.10). Electrical current through a solenoid on the motor controls the motor’s displacement. When configured for driving, the torque accelerates the vehicle and fluids flows from the high pressure side of the system into the low pressure reservoir. When configured for regenerative braking, the torque opposes the motion of the vehicle and fluid is drawn from the low pressure reservoir and forced into the high pressure side of the system. Unlike the pump, the motors do require an external pilot pressure to operate. This pressure, which must remain between 30 and 100bar, is supplied by a pressure reducing valve common to both motors.

2.4 Energy Storage

The team's performance goals for the vehicle also show in the energy storage system where four 10 gallon bladder accumulators are used. At 40 gallons, this is significantly more than the 56.8 liters of storage in the EPA system and slightly more than Buchwald et al. specified for a city bus. This system was sized to provide enough energy for a 1/4 mile run on a dragstrip. Using four discrete accumulators rather than a single 40 gallon accumulator gives the team the ability to run the vehicle with an accumulator volume in increments of 10 gallons by removing one or more accumulators from the system.

2.5 Flow Control

Poppet type slip-in cartridge valves in an H-bridge configuration enclosed in a manifold control the fluid flow path for the motors. Each motor has its own H-bridge manifold. The H-bridges allow the motors to provide positive and negative torque in both the forward and reverse directions. Figure 2.3 shows the valve states in the h-bridge for motoring and regenerative braking for one direction of travel. For the opposite direction of travel, the valve states are inverted. The valve states for freewheeling do not change with respect to direction of travel.

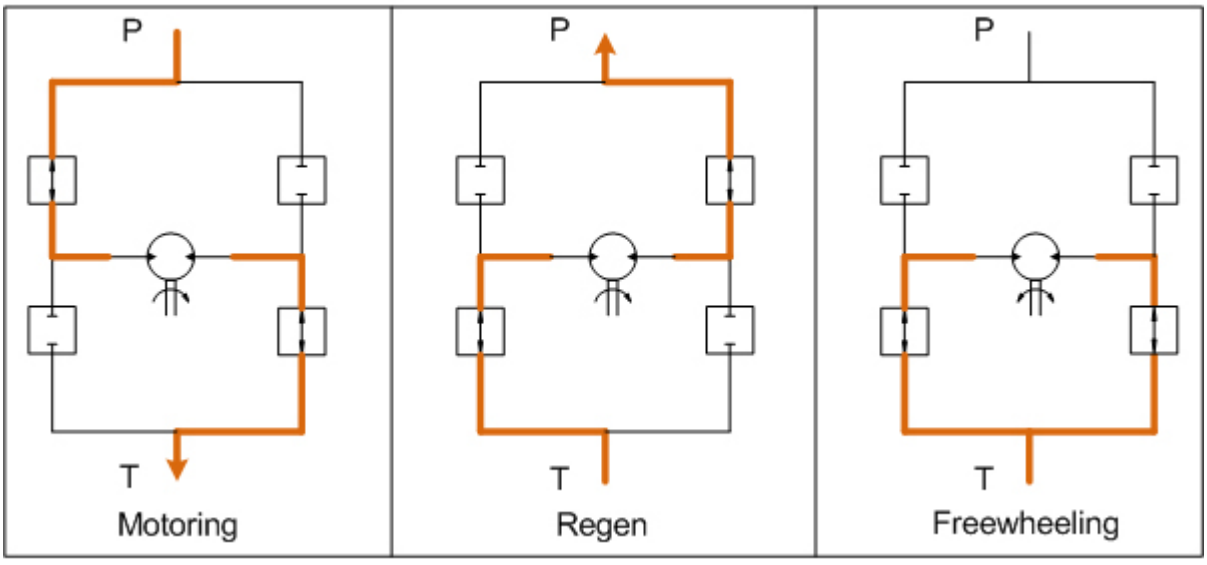


Figure 2.3: H-Bridge valve configuration for motoring, regen, and freewheeling

The cartridge valves are normally closed and held in this state by a return spring, internal to the valve, when the system is depressurized. When the system is pressurized, the valves are controlled via a pilot port in the valve cap. The pilot for each valve is controlled by an electronically actuated 3-way directional control valve. The directional control valve's default position connects the pilot to the high pressure side of the system ensuring the cartridge valves have a normally closed operation. This is safety feature that prevents the motors from producing torque unless the valves are electronically commanded to open. A pressure reducing valve set to 350bar supplies fluid to the directional control valves at their rated pressure. This valve arrangement is shown in Figure 2.4 and uses the same line designation as the complete hydraulic schematic. A lower pilot pressure than system pressure is acceptable for the cartridge valves due to the area ratio between the pilot surface, C, and the flow surfaces, A and B. At full pressure, 400bar, the 0.4:1 area ratio between the B and C surfaces allow for a safety factor of 2.2. At anything below full pressure, the safety factor is higher. Poppet valves were selected to minimize leakage.

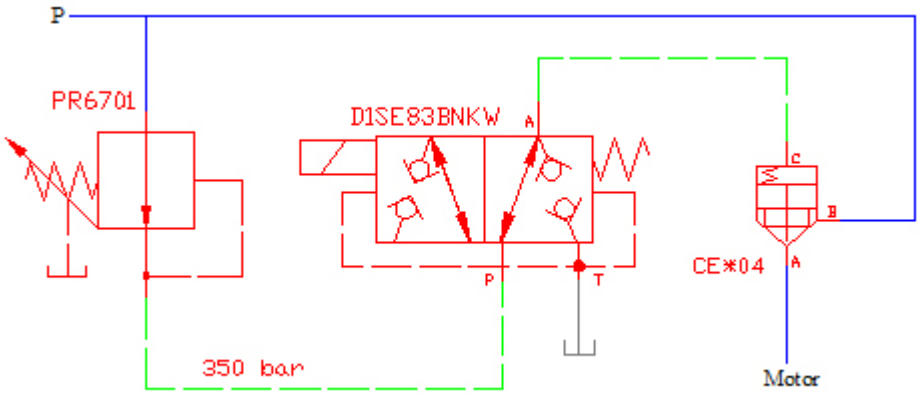


Figure 2.4: Cartridge valve control circuit

2.6 Fluid Conditioning

The fluid conditioning system provides temperature and fluid cleanliness management. Most of this system is contained in the kidney loop which circulates fluid from the low pressure reservoir on an as needed basis. The circuit is controlled electronically with a 3-way directional control valve. When activated, high pressure fluid is supplied to a small hydraulic motor which drives a larger hydraulic pump.

A tube and shell heat exchanger removes heat from the fluid after it leaves the pump. Fluid in the kidney loop also passes through a coarse filter to maintain the cleanliness of the fluid to the level specified by the pump and motor documentation before entry to the motors while they are providing regenerative braking and the pump.

In many cases the filtration specifications vary throughout the system. To ensure proper and reliable operation of the hydraulic system, fluid cleanliness must meet or exceed the levels specified in the documentation of the components. In order to achieve this, this system utilizes a second filtration element after the pump to meet the criteria specified in Parker's Handbook of Hydraulic Filtration. The system's filtration requirements for the pump and motors are significantly lower than that required by the valves. The post pump filter is connected directly to the output of the pump which is the path the majority of the fluid flowing through the hydraulic system will follow. The post pump filter is finer than the one in the kidney-loop and further reduces the level of contaminants in the hydraulic fluid to the lowest level specified in the component datasheets.

2.7 Safety

Safety is of primary importance in a high pressure hydraulic system. Careful planning is necessary to avoid personal injury and damaging the equipment. The hydraulic lines, connections, and components should be inspected regularly to prevent hydraulic injection injuries from pinhole leaks when the system is pressurized. Before any maintenance is performed, the system must be completely depressurized using the bleed valve connected to the accumulator manifold. The ball valve connecting the accumulators to the system must be open during depressurization otherwise parts of the system may retain pressure. This valve should remain closed when the vehicle is not in use to unload the system and prevent leakage.

Component specifications for pressure must not be exceeded. A pressure relief valve set for the maximum allowable operating pressure bleeds off excess pressure from the system. In the event that the system’s relief valve is unable to prevent the system pressure from rising, rupture discs attached to the gas side of the accumulators will burst, allowing the nitrogen to escape, and completely depressurize the system.

Control valve states and switching times are also cause for concern. If the cartridge valves fail to operate as intended and there is not a valve open on each side of the motor while it is rotating and has a displacement greater than zero, a very hazardous situation would be created. Pressure can build quickly on the discharging side of the motor causing an overpressure or burst. On the suction side of the motor, cavitation can cause damage to the motor. Check valves, shown in Figure 2.5, are placed on each leg of the h-bridge to prevent these scenarios. When the pressure differential across the valve exceeds their seating pressure the valves open and allow flow.

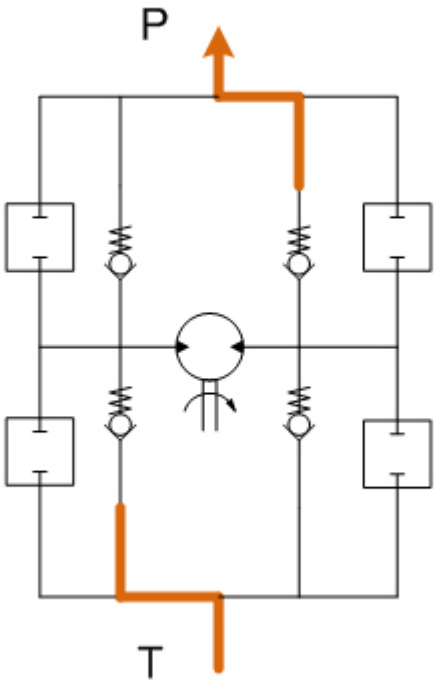


Figure 2.5: H-bridge check valve pressure relief

Chapter 3: **Component Models**

Many factors contribute to a vehicle's fuel economy, each of which must be modeled appropriately to produce an accurate simulation. Each component in the hydraulic system has an associated efficiency. Some of these losses are rather small and can be neglected, such as the leakage flow through a poppet valve. However, losses due to mechanical and volumetric efficiencies of the motor and pump can be up to 25% for each component at full displacement. Data at less than full displacement is not generally available and must be modeled to maintain the simulation's accuracy. The pressure-volume-temperature relation of the gas spring in the energy storage system affects the pressure profile the hydraulic components see as well as the energy storage capacity of the accumulator. The pressure profile is important as the motor and pump efficiencies are pressure dependent. An Engine's fuel efficiency varies significantly throughout its operating range. This data is also not commonly available; however, the equipment to measure the engine's performance was available so this efficiency was measured rather than modeled. Road load, the combination of wind resistance and rolling resistance, represents a large loss. This data is measured by vehicle manufacturers and fit to a quadratic equation. Losses through gearboxes are small but easily modeled with a constant.

3.1 Pump

An axial piston pump can be modeled by Equation 3.1 and Equation 3.2 when neglecting dynamic effects (Manring, 2005).

$$T = \frac{\Delta P \mathrm{V}}{\eta_{\mathrm{mp}}} \tag{3.1}$$

$$Q = \omega V \eta_{vp} \tag{3.2}$$

Equation 3.1 relates the torque input to the pump to the pressure differential across the pump's ports, the displaced volume, and the mechanical efficiency. The low pressure reservoir is assumed to be unpressurized and at 0 psig. ΔP can therefore be replaced by P, the pressure of the high pressure side of

the system. Equation 3.2 relates the fluid flow rate into the high pressure side of the system to the input shaft speed, displaced volume, and the volumetric efficiency.

The volumetric efficiency for a pump is defined in Equations 3.3 where Q_l is the leakage flow (Manring, 2005).

$$\eta_{vp} \equiv \frac{\omega V - Q_l}{\omega V} \tag{3.3}$$

Since the pump in this system is a variable displacement unit, this functionality must be included in Equations 3.2 and 3.3. First, the percent of maximum displacement, θ, is defined in equation 3.4 where V_d is the current displacement of the pump and V_{max} is the maximum pump displacement.

$$\theta \equiv \frac{V_d}{V_{max}} \tag{3.4}$$

Next, the leakage paths are assumed to be unaffected by changes in displacement and therefore the volumetric efficiency is constant with respect to displacement. Volumetric efficiency data is only reported at maximum displacement in the datasheets and this allows performance to be predicted at smaller displacements.

$$Q_l = \omega V_{max}(1 - \eta_{vp\,max}) \tag{3.5}$$

Substituting Equation 3.5 into Equation 3.3 produces Equation 3.6, the volumetric efficiency of a pump as a function of displacement.

$$\eta_{vp} = \frac{\omega V_d - \omega V_{max}(1 - \eta_{vp\,max})}{\omega V_d} \tag{3.6}$$

Substituting Equation 3.4 into Equation 3.6 and simplifying yields Equation 3.7.

$$\eta_{vp} = 1 - \frac{(1 - \eta_{vp\,max})}{\theta} \tag{3.7}$$

Substituting Equation 3.7 into 3.2 gives the volumetric flow rate of a pump as a function of displacement.

$$Q = \omega V_d \left[1 - \frac{\left(1 - \eta_{vp\,max}\right)}{\theta}\right] \quad 3.8$$

$$T = \frac{PV_d}{\eta_{mp}} \quad 3.9$$

Equations 3.8 and 3.9 are used to describe the pump in this system. The mechanical and volumetric efficiencies are also functions of shaft speed and pressure. These efficiencies are calculated at full displacement using the pump's datasheet which creates a 2-D array (Denison Hydraulics, 2003). This array is then fed into the griddata function in MATLAB to create a finer mesh of data. Finally, the 2-D volumetric efficiency array is used in conjunction with equation 3.7 to create a 3-D array with axes of speed, pressure, and displacement.

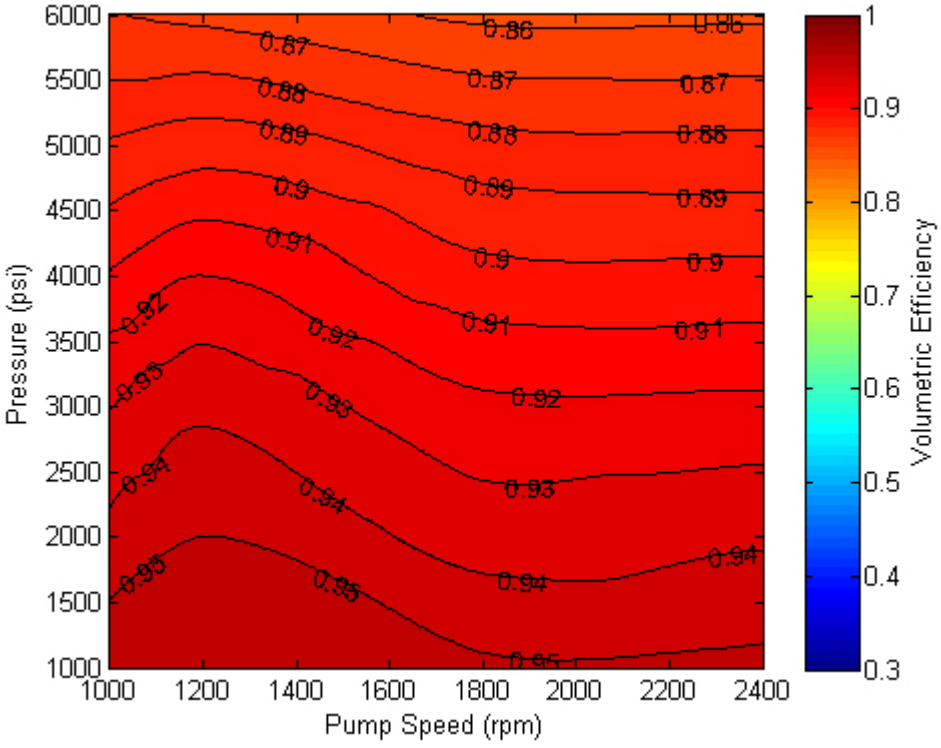


Figure 3.1: Pump volumetric efficiency map at full displacement

3.2 Motor

A bent-axis motor can be modeled Equations 3.10 and 3.11 (Manring, 2005).

$$T = \Delta P V \eta_{mm} \tag{3.10}$$

$$Q = \frac{\omega V}{\eta_{vm}} \tag{3.11}$$

Equation 3.10 relates the torque output of the pump to the pressure differential across the motor's ports, the displaced volume, and the mechanical efficiency. The low pressure reservoir is assumed to be unpressurized and at 0 psig. ΔP can therefore be replaced by P, the pressure of the high pressure side of the system. Equation 3.11 relates the volumetric flow rate through the motor to the shaft speed, volumetric displacement, and the mechanical efficiency.

The volumetric efficiency of a motor is defined in Equation 3.12 (Manring, 2005).

$$\eta_{vm} \equiv \frac{\omega V}{\omega V + Q_l} \tag{3.12}$$

Since the datasheet for the motor only provide data at full displacement, the efficiencies at smaller displacements must be approximated. To accomplish this, the leakage flow is assumed to be constant with respect to displacement. In Equation 3.13 the leakage flow is solved for in terms of the volumetric efficiency at full displacement.

$$Q_l = \omega V_{max} \left(\frac{1}{\eta_{vm\,max}} - 1 \right) \tag{3.13}$$

Substituting Equation 3.13 into Equation 3.12 gives the volumetric efficiency of a motor as a function of displacement.

$$\eta_{vm} \equiv \frac{\omega V_d}{\omega V_d + \omega V_{max} \left(\frac{1}{\eta_{vm\,max}} - 1 \right)} \tag{3.14}$$

Substituting Equation 3.4 into 3.14 and simplifying yields Equation 3.15.

$$\eta_{vm} = \frac{1}{1 + \frac{1}{\theta}\left(\frac{1}{\eta_{vm\,max}} - 1\right)} \quad 3.15$$

Substituting Equation 3.15 into Equation 3.11 gives the volumetric flow rate of the motor as a function of displacement.

$$Q = \omega V_d \left[1 + \frac{1}{\theta}\left(\frac{1}{\eta_{vm\,max}} - 1\right)\right] \quad 3.16$$

$$T = P\mathrm{V}\eta_{\mathrm{mm}} \quad 3.17$$

Equations 3.16 and 3.17 are used to describe the motor while it is motoring. As with the pump, the mechanical and volumetric efficiencies are functions of shaft speed and pressure. The minimum shaft speed of the data provided in the motor's datasheet is approximately 260 rpm (Denison Hydraulics, 2003). Since the motor will have to operate at lower speeds than this as the vehicle comes to and leaves from a stop, efficiencies for shaft speeds below 260 rpm had to be extrapolated. The flow rate with respect to shaft speed is very linear, as shown in Figure 3.2, and the assumption that this trend continues to 0 shaft speed is made. Flow rate data is shown for both pressures given on the motor's data sheet. The mechanical efficiency is fitted with a quadratic equation and extrapolated to 0 rpm. A sample of the mechanical efficiency data is shown in Figure 3.3. 2-D arrays are made using the efficiency data and are then put into the griddata function in MATLAB to create a finer grid of data. The 2-D volumetric efficiency array is used with Equation 3.15 to make a 3-D array of volumetric efficiency with axes of shaft speed, pressure, and displacement. The motor's volumetric efficiency at full displacement is shown in Figure 3.4.

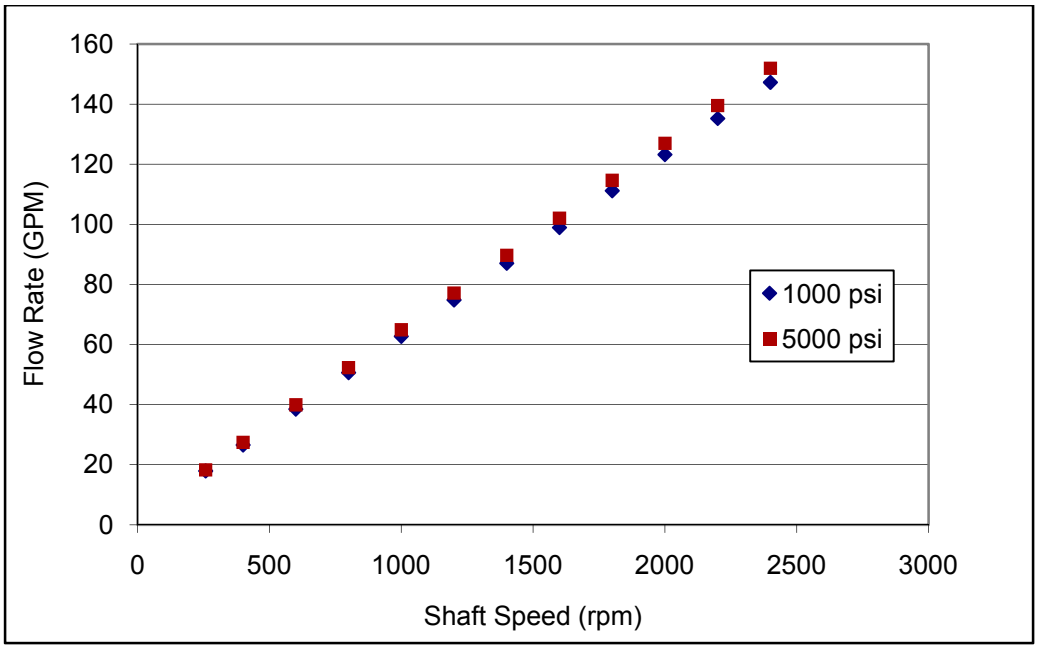


Figure 3.2: Motor flow rate with respect to shaft speed and pressure

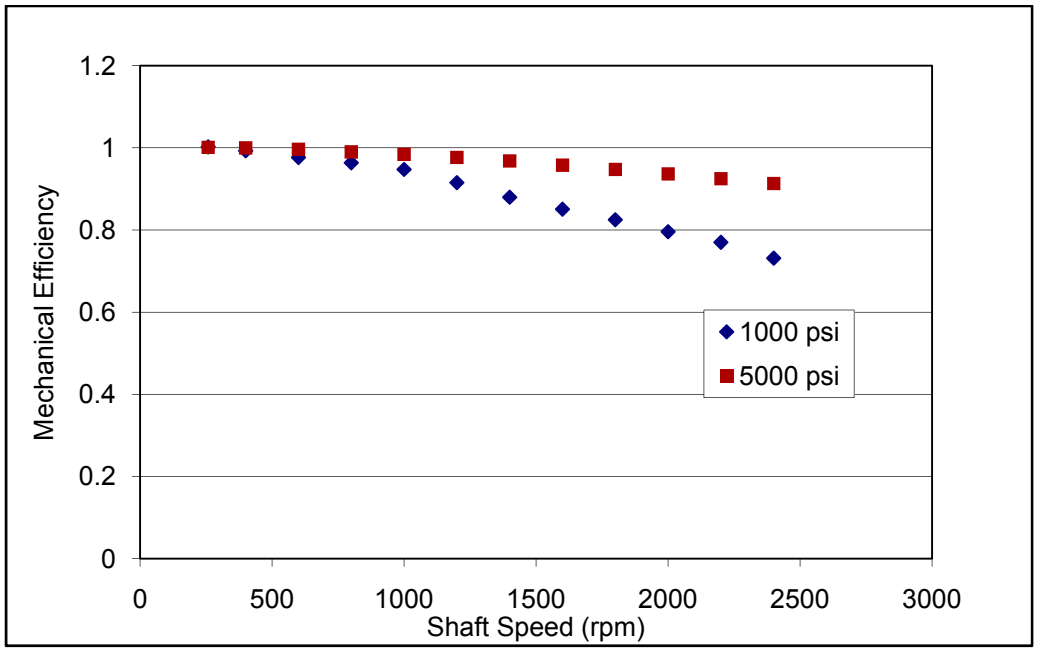


Figure 3.3: Motor mechanical efficiency with respect to shaft speed and pressure

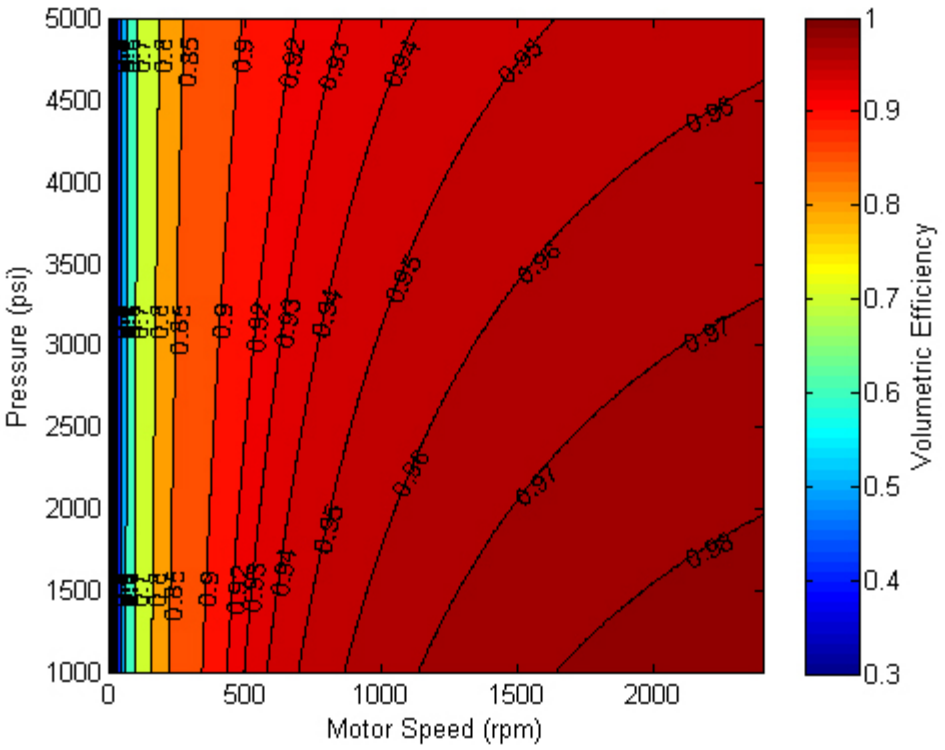


Figure 3.4: Motor volumetric efficiency at full displacement

When the motor is acting as a pump during regenerative braking the mechanical and volumetric efficiencies from motoring are used. The motoring volumetric efficiency can be turned into a pumping volumetric efficiency if the leakage flow rate is assumed to be the same during motoring and pumping. This assumption provides normal looking values when the motoring volumetric efficiency is large; however, as the motoring volumetric efficiency starts to drop off, the predicted pumping volumetric efficiency becomes unrealistically low. A motoring volumetric efficiency of 50% would yield a pumping volumetric efficiency of 0%. This can be shown by solving equations 3.3 and 3.12 for the leakage flow, equating them, and then solving for the pumping volumetric efficiency.

$$Q_l = \omega V(1 - \eta_{vp}) = \omega V\left(\frac{1}{\eta_{vm}} - 1\right) \qquad 3.18$$

$$\eta_{vp} = 2 - \frac{1}{\eta_{vm}} \quad 3.19$$

3.3 Accumulator

The nitrogen in the accumulator is modeled using the Benedict-Webb-Rubin equation of state (Equation 3.20). This equation has been shown to be much more accurate than the ideal gas law in the range of operation of hydraulic accumulators and extremely close to data published by the National Bureau of Standards (Pourmovahed, Beachley, & Fronczak, Modeling of a Hydraulic Energy Regeneration System - Part I: Analytical Treatment, 1992a). Constants for the Benedict-Webb-Rubin equation are shown in Table 3.1. Values for the specific volume of nitrogen are interpolated from a table with axes of temperature and pressure (Little & A, 1962). Nitrogen's constant volume specific heat is allowed to vary with temperature and pressure. Along with the nitrogen, an elastomeric open-cell foam resides in the gas side of the accumulator. Adding an open-cell foam with a high specific heat to the gas side of the hydraulic accumulator significantly reduces the change in temperature due to the compression and expansion of the nitrogen. The mass of foam was determined by using the same foam mass to gas volume ratio as presented by Pourmovahed. The energy equation for the nitrogen-foam system is shown in Equation 3.21. The heat loss term has been neglected due to uncertainties in obtaining the quantity. A fundamental equation for the thermodynamic properties of nitrogen, explicit in Helmholtz energy, was used along with Equation 3.22 to determine the constant volume specific heats of nitrogen for the range of temperatures and pressures corresponding to the use of a hydraulic accumulator (Jacobsen, Stewart, & Jahangiri, 1986). This generates a table of values that is interpolated from.

The hydraulic accumulator model is implemented by first using the precharge pressure and temperature specified in the configuration file to find the specific volume of the nitrogen. Dividing the maximum gas volume of the hydraulic accumulator by the specific volume yields the mass of nitrogen. Initial conditions for the accumulator are set by a temperature and pressure specified in the configuration file and an interpolated value for specific volume. At each time step, the net fluid flow into the accumulator is used

to compute a new gas volume and specific volume. A new value for the constant volume specific heat is calculated using the temperature and pressure from the previous time step. The energy equation is then integrated for the time step and used to find the current temperature of the nitrogen. Once the temperature is known, the Benedict-Webb-Rubin equation of state is used to determine the current system pressure.

$$p_g = \frac{RT}{v} + \frac{B_0 RT - A_0 - \frac{C_0}{T^2}}{v^2} + \frac{bRT - a}{v^3} + \frac{a\alpha}{v^6} + \frac{C\left(1 + \frac{\gamma}{v^2}\right)e^{\frac{-\gamma}{v^2}}}{v^3 T^2} \tag{3.20}$$

$$\left(1 + \frac{m_f c_f}{m_g c_v}\right)\frac{dT}{dt} = -\frac{1}{c_v}\left[\frac{RT}{v}\left(1 + \frac{b}{v^2}\right) + \frac{B_0 RT + \frac{2C_0}{T^2}}{v^2} - \frac{2\left(1 + \frac{\gamma}{v^2}\right)e^{\frac{-\gamma}{v^2}}}{v^3 T^2}\right]\frac{dv}{dt} \tag{3.21}$$

Table 3.1: Benedict-Webb-Rubin Constants

Constant	Value	Units
a	1.15712E-01	$(m^3/kg)^3$ Pa
A_0	1.360430E+02	$(m^3/kg)^2$ Pa
b	2.96688E-06	$(m^3/kg)^2$
B_0	1.454573E-03	(m^3/kg)
c	3.357732E+03	$(m^3/kg)^3$ K^2Pa
C_0	1.0405534E+06	$(m^3/kg)^2$ K^2Pa
α	5.7882567E-09	$(m^3/kg)^3$
γ	6.755378E-06	$(m^3/kg)^2$
R	2.967551E+02	(m^3/kg) Pa/K

$$C_v = -R\tau^2\left(\frac{\partial^2 \alpha^0}{\partial \tau^2} + \frac{\partial^2 \bar{\alpha}}{\partial \tau^2}\right) \tag{3.22}$$

3.4 Engine

Performance data from a Perkins 1006-6TW 6-cylinder turbo diesel was selected for this simulation due to its availability and the instrumentation attached to it. The 1.9L VW TDI used in the HVDT project was not instrumented and complete performance maps were not available. A complete efficiency map was created for the Perkins using an engine dynamometer, to measure engine speed and torque output, and a

fuel balance, to measure fuel flow rate. Data was collected from idle (1000rpm) to the speed limiter (2500rpm) and from 50lbf-ft to approximately 450lbf-ft. Measurements below 50lbf-ft were not possible due to large fluctuations in engine output. Efficiency data below 50lbf-ft were extrapolated using a cubic fit. Engine efficiency is measured by brake specific fuel consumption (BSFC) which is the mass flow rate of fuel divided by the power output at the output shaft. This calculation is shown in equation 3.23 using the fuel flow rate (FFR), torque output, and the engine speed. Figure 3.5 shows the complete engine map for the Perkins.

$$BSFC\left(\frac{lbm}{hp\ hr}\right) = \frac{36300\ FFR(kg/hr)}{\pi\ T(lb_f - ft)\ \omega(rpm)} \tag{3.23}$$

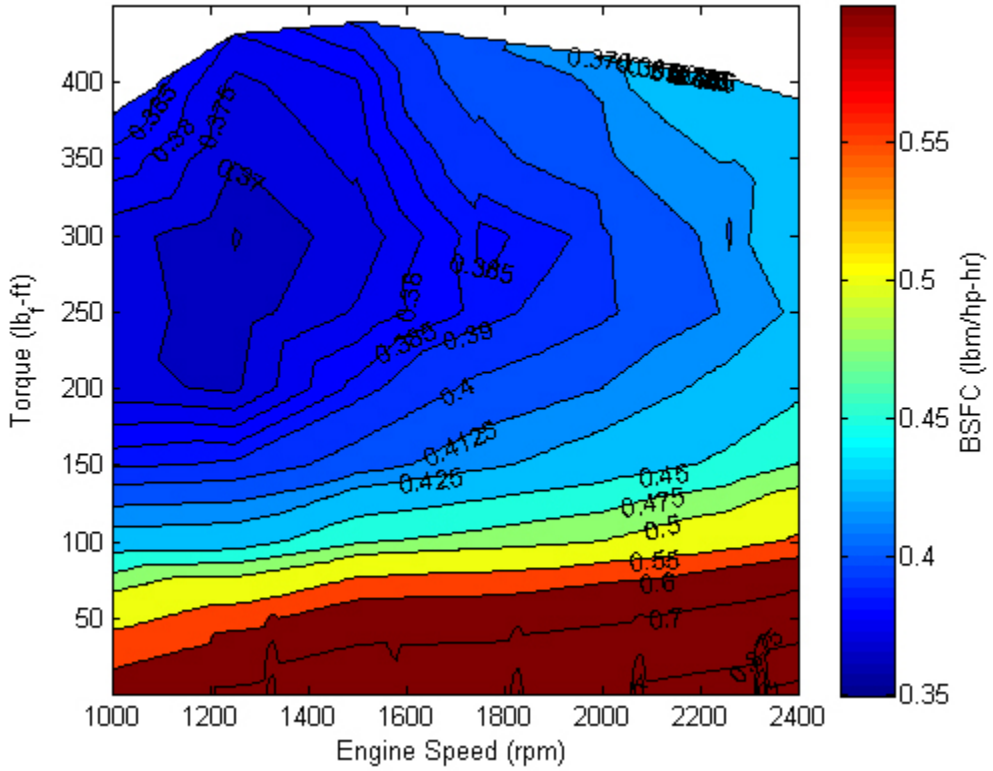


Figure 3.5: Perkins BSFC map

3.5 Road Load

Road load is calculated using averaged manufacturer supplied road load data for large SUVs show in Table 3.2 (Alson, et al., 2004). The coefficients in Table 3.2 are used with Equation 3.24 to determine the road load, F_{rl}, at a given velocity, V. The average force at each time step is found by integrating the road load from the current velocity to the projected velocity at the next time step and then dividing by the projected change in velocity.

Table 3.2: Road load coefficients

A (lbf)	B (lbf/mph)	C (lbf/mph^2)
56.69	1.1514	0.03121

$$F_{rl} = CV^2 + BV + A \tag{3.24}$$

$$F_{rl} = \frac{\int_V^{V+at}(CV^2 + BV + A)dV}{at} \tag{3.25}$$

3.6 Gearbox

HVDT's choice to not include multispeed gearboxes was made for cost and construction reasons which do not apply to creating a model of the vehicle. Since the efficiencies of the components that would be connected to gearboxes are dependent upon speed, multispeed gearboxes were included between the engine and pump and also between the motor and differential. Gearbox efficiency is considered to be a constant 99% based on a gearbox tested by Martins et al.

3.7 Simplifications

Many simplifications were made to the overall vehicle model in order limit the complexity of the simulation. A major difference between the HVDT vehicle and the model is the lack of four wheel drive in the model. Since the model is not currently focused on off road simulation, the four wheel functionality is not needed at this time. Dynamic effects of the components are neglected. This includes engine, pump, and motor inertias, turbo lag, rates of change of displacement, and valve actuation times. Pressure losses

through the manifolds, valves, and hoses are also neglecting. The effect of these quantities is considered to be small and would add significant complexity and computational requirements to the simulation.

Chapter 4: **Simulation**

4.1 Operational Modes

Control of the engine and hydraulic system during the simulation is categorized into four operational modes: hybrid, hydrostatic, regeneration, and idle. Hybrid mode uses the hydraulic system in the series powertrain configuration where the accumulator is connected to the hydraulic system and the engine's output is decoupled from the immediate power requirements. Hydrostatic mode disconnects the accumulator and uses the hydraulic system as a hydrostatic transmission. The engine's output is not decoupled for the immediate power requirements in this mode. Regeneration mode uses the motor as a pump to reclaim the vehicle's kinetic energy while decelerating and stores it in the accumulator. The regeneration mode is able to operate concurrently with hybrid mode but not hydrostatic mode. Idle mode controls the engine when power from the engine is not required.

4.1.1 Hybrid

Hybrid mode is operated using an upper and lower pressure threshold. The lower pressure threshold is the precharge pressure plus a buffer pressure. The buffer pressure is a safety device to prevent cavitation should the motor's demand for fluid exceed the amount of fluid in the system. This can occur if the pump is unable to match the flow rate of the motor while the pressure is low. While the hybrid mode is enabled, the valve connecting the accumulators to the rest of the system is open. Figure 4.1 shows the fluid flow path while in hybrid mode. When the system pressure drops below the lower threshold, the system begins to charge. The system stops charging when the system pressure exceeds the upper threshold.

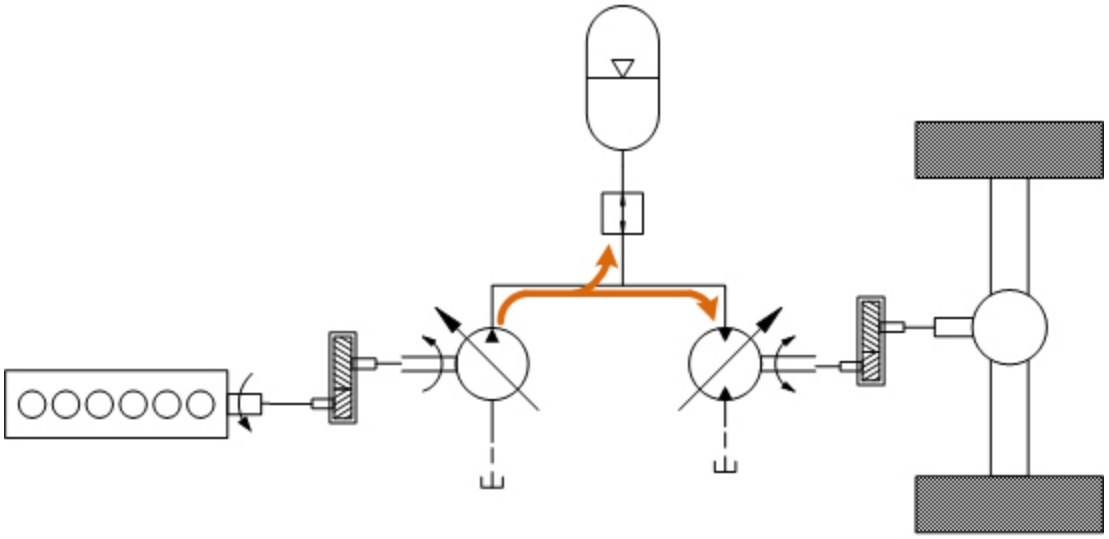


Figure 4.1: Hybrid mode fluid flow path

The engine and pump parameters - engine speed, engine torque, pump speed, pump torque, and pump displacement - are based on the system's pressure. At a given pressure, the combined engine and pump system has a peak efficiency. The peak efficiency at this point in the system will be referred to as the minimum effective BSFC at the pump. The effective BSFC at the pump ($eBSFC_P$) is the engine's BSFC divided by the overall pump efficiency and gearbox efficiency (Equation 4.1). Along with pressure, the $eBSFC_P$ is a function of shaft speed, torque, pump displacement, and gear ratio. Using Equation 3.9, the pump's torque input, shaft speed, and mechanical efficiency can be substituted for its displacement. This leaves the effective BSFC as a function of shaft speed, torque, pressure, and gear ratio (Equation 4.2). The BSFC and mechanical efficiency arrays are replicated along the pressure axis to form 3-D arrays. Element-wise operations are performed to create a 4-D array for the $eBSFC_P$. At a given pressure and gear ratio, the minimum effective BSFC can be found with respect to shaft speed and torque. This yields a 2-D array with all necessary parameters to define the operation of the combined engine and pump system. The gear ratio with the smallest BSFC at the current pressure is selected for use. Figure 4.2 is a graphical description of this process. The cube represents the 4-D $eBSFC_P$ array with the 4th dimension represented

by the 1-D array pointing into it. Elements in the arrays contain the data listed in the cells pointing at them. Large arrows indicate the path the method follows.

$$eBSFC_P = \frac{BSFC}{\eta_v \eta_m \eta_G} \tag{4.1}$$

$$T = \frac{PV_d}{\eta_{\mathrm{mp}}} \tag{3.9}$$

$$eBSFC_P = \frac{BSFC(\omega, T)}{\eta_v \left(\omega N_G, P, \frac{T}{N_G}\right) \eta_m(\omega, P) \eta_G} \tag{4.2}$$

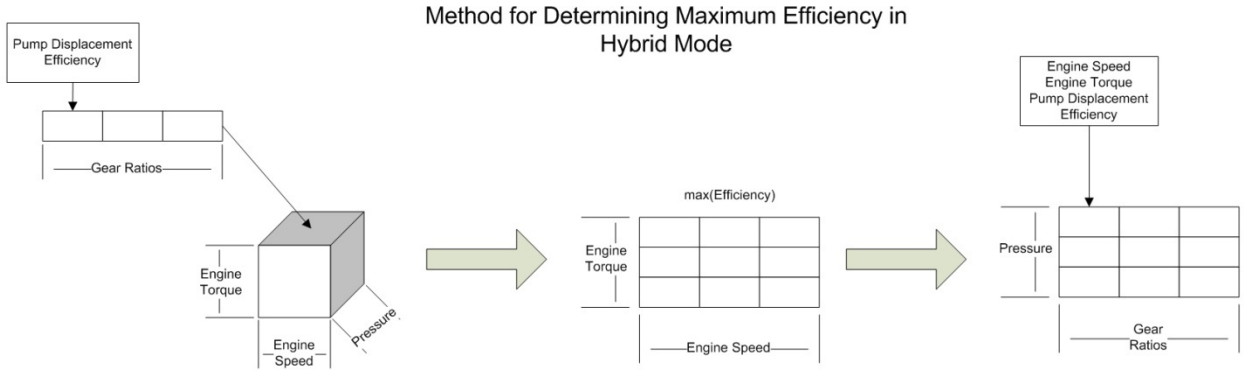


Figure 4.2: Program method for determining system parameters for maximum efficiency during hybrid mode

4.1.2 Hydrostatic

In hydrostatic mode, the system is configured to act like a hydrostatic transmission. The accumulator is disconnected from the system as shown in Figure 4.3.

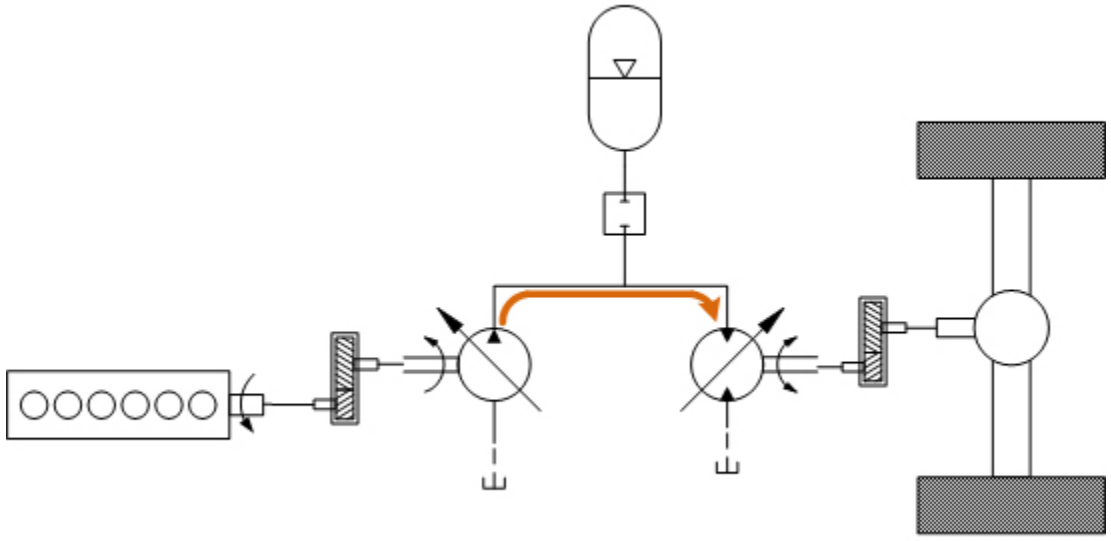


Figure 4.3: Hydrostatic mode fluid flow path

System parameters in this mode are based on the vehicle's velocity and the torque requirement. This mode's peak efficiency, minimum effective BSFC at the motor, is found using the vehicle speed and torque requirement. Effective BSFC at the motor ($eBSFC_M$) is the engine's BSFC divided by the overall pump and motor efficiencies. In hydrostatic mode, $eBSFC_M$ is a function of engine speed, engine torque, engine-pump gear ratio, motor speed, motor torque, and motor-differential gear ratio. The process for finding the minimum $eBSFC_M$ begins at the motor. For a given motor speed and torque output, there are a range of pressures and displacements that will satisfy the torque requirement. This means that there is a vector of flow rates for each motor speed and torque combination. The elements in this 3-D array contain pressure, motor volumetric efficiency, and motor mechanical efficiency. Since leakage and pressure drops in the system are neglected, these quantities must be the same at the pump and the motor. For each flow rate and pressure there are range of pump speeds and displacements that satisfy the requirements in each

engine-pump gear ratio. The minimum $eBSFC_P$ is found for each engine-pump gear at each flow rate and pressure. This creates a 3-D array with axes of flow rate, pressure, and engine-pump gear ratio whose elements contain engine speed, engine torque, and pump displacement. The 3-D array from the pump is combined with the 3-D array from the pump along their flow rate axis creating a 4-D array with axes of motor torque, motor speed, flow rate, and engine-pump gear ratio. The minimum $eBSFC_M$ is found along the flow rate axis reducing the array to 3-D array. This array's elements contain engine speed, engine torque, pump displacement, pump efficiencies, pressure, flow rate, motor speed, motor torque, and motor efficiencies. A graphical description of this method is shown in Figure 4.4.

$$eBSFC_M = \frac{BSFC}{\eta_{vp}\eta_{mp}\eta_G\eta_{vm}\eta_{mm}} \tag{4.3}$$

When this mode is activated, the motor speed and torque are calculated for each motor-differential gear. Then the minimum $eBSFC_M$ is found along the engine-pump gear ratio axis for each motor-differential gear ratio. Finally, the motor-differential gear with the lowest $eBSFC_M$ is found and the system parameters are fully defined.

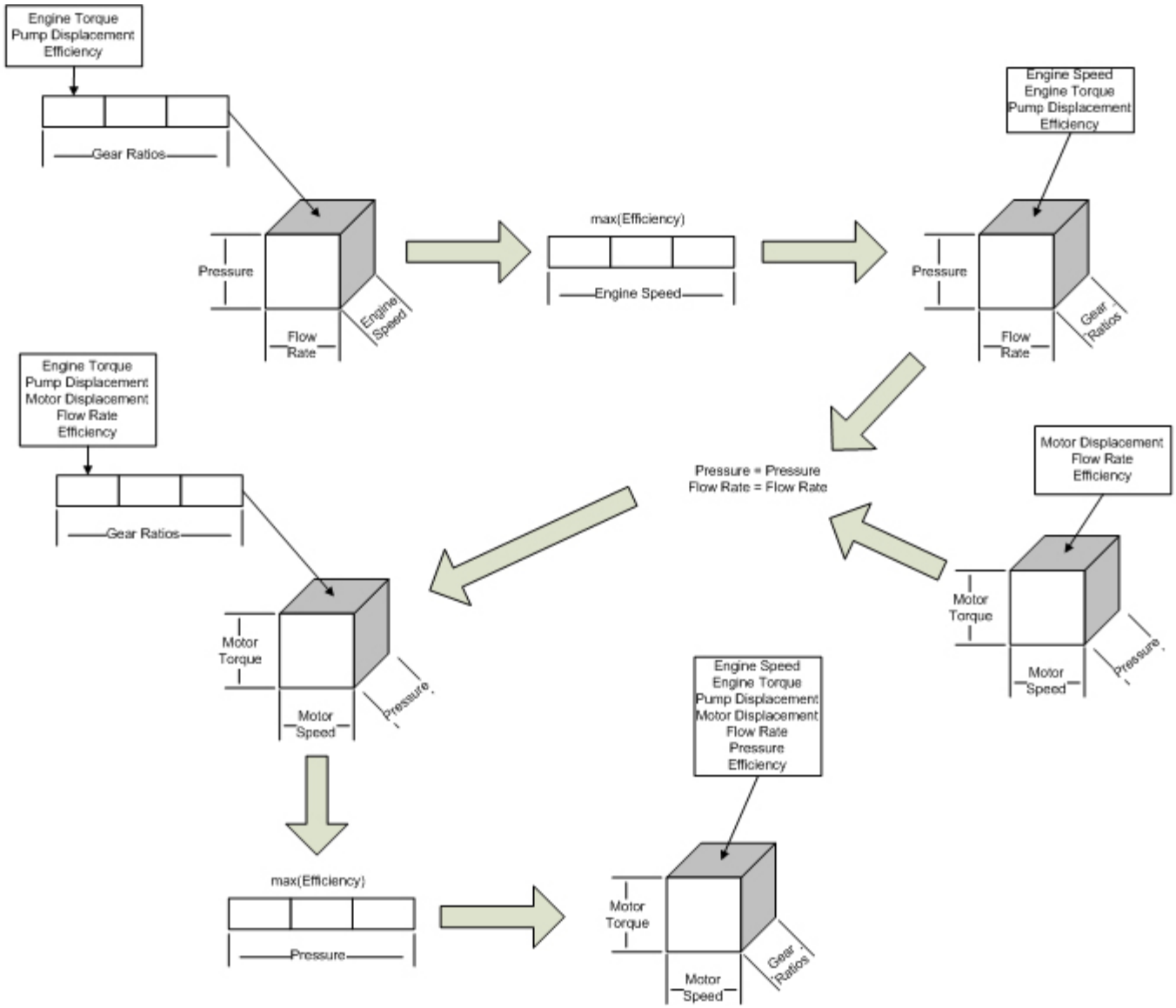


Figure 4.4: Program flow chart for determining system parameters for maximum efficiency during hydrostatic mode

4.1.3 Regeneration

In the regenerative braking mode, the vehicle's kinetic energy is converted into potential energy and stored in the accumulators. The fluid flow path is shown in Figure 4.5.

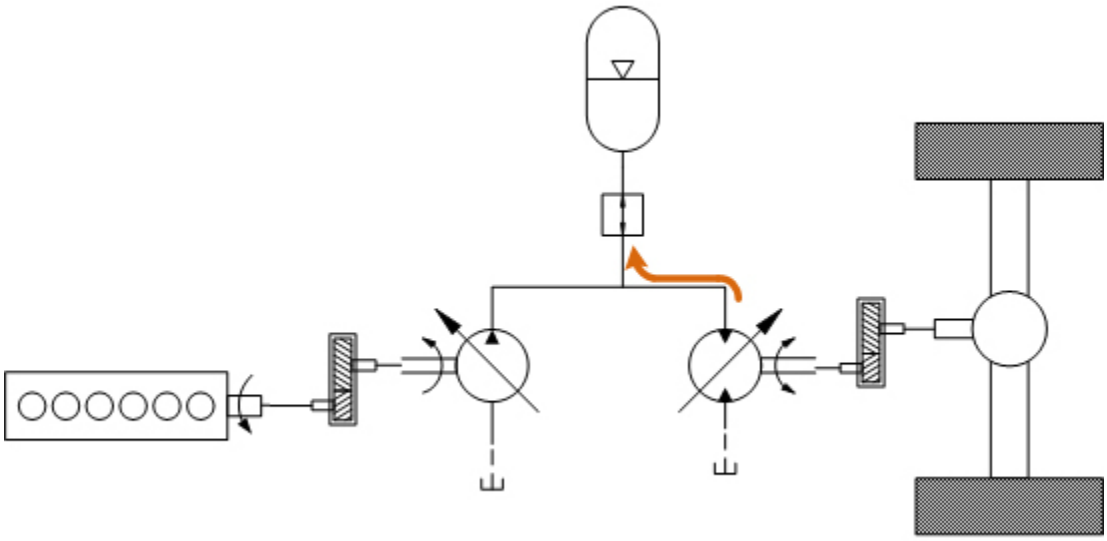


Figure 4.5: Regeneration mode fluid flow path

When this mode is activated, the motor speed and torque is calculated for each motor-differential gear ratio. Using these values with the system pressure allows the motor efficiency to be calculated in each motor-differential gear. These values are calculated starting with the lowest gear. If the motor would be operating above its speed range, the gear is thrown out. If the system pressure is too low, the motor will not be able to provide the required torque. The motor displacement for the gear ratio is set to full when this happens. If the required torque can not be produced in any of the gears, the lowest gear ratio that does not overspeed the motor is selected to maximize the amount of reclaimed energy and the mechanical brakes are used to make up the balance of torque. When the required torque can be produced, the gear with the greatest motor efficiency is chosen.

4.1.4 Idle

Idle mode is activated when power from the engine is not needed. This mode can be configured to let the engine idle or shut down the engine. The Perkins uses 1.1 kg/hr of fuel while it is idling.

4.2 Strategy

The default control strategy implemented in the simulation attempts to run the vehicle in the most efficient state for the current time step. This is done by comparing the $eBSFC_M$ for the hybrid and hydrostatic modes. The mode with the lowest $eBSFC_M$ is selected for the timestep. Hydrostatic mode's $eBSFC_M$ is interpolated from its 3-D array. Since the accumulator decouples the engine-pump system's output from the motor, the $eBSFC_M$ can not be calculated in the same way for the hybrid mode. Hybrid mode's $eBSFC_M$ is still the $eBSFC_P$ divided by the motor's efficiency. However, the $eBSFC_P$ is calculated by keeping track of an energy averaged $eBSFC_P$ for the energy stored in the accumulator as shown in Equation 4.4. The energy averaged $eBSFC_P$ is then divided by the motor efficiency for each motor-differential gear ratio. The smallest of these values is compared to the $eBSFC_M$ from the hydrostatic mode to determine which mode is used for the time step. In the event that the system pressure is not sufficient to meet the torque requirement in the hybrid mode, the hydrostatic mode is selected. If the system pressure is lower than the lower threshold for hybrid mode and the pump can supply enough fluid to at least maintain the pressure, the hybrid mode is selected if it is more efficient that the hydrostatic mode. If the net flow into the hydraulic system becomes negative while the pressure is below the threshold, the hydrostatic mode is activated. When energy from regenerative braking is available, its use is prioritized ahead of hydrostatic mode. The engine is shut off in the idle mode to conserve fuel.

$$eBSFC_{PAVG}(N) = \frac{eBSFC_{PAVG}(N-1)E_{total}(N-1) + eBSFC_P(N)E(N)}{E_{total}(N-1) + E(N)} \quad 4.4$$

Note that the regeneration mode can be enabled while the hybrid or idle modes are enabled. The default strategy can be altered by disabling modes using Boolean values and by changing parameters in the configuration file.

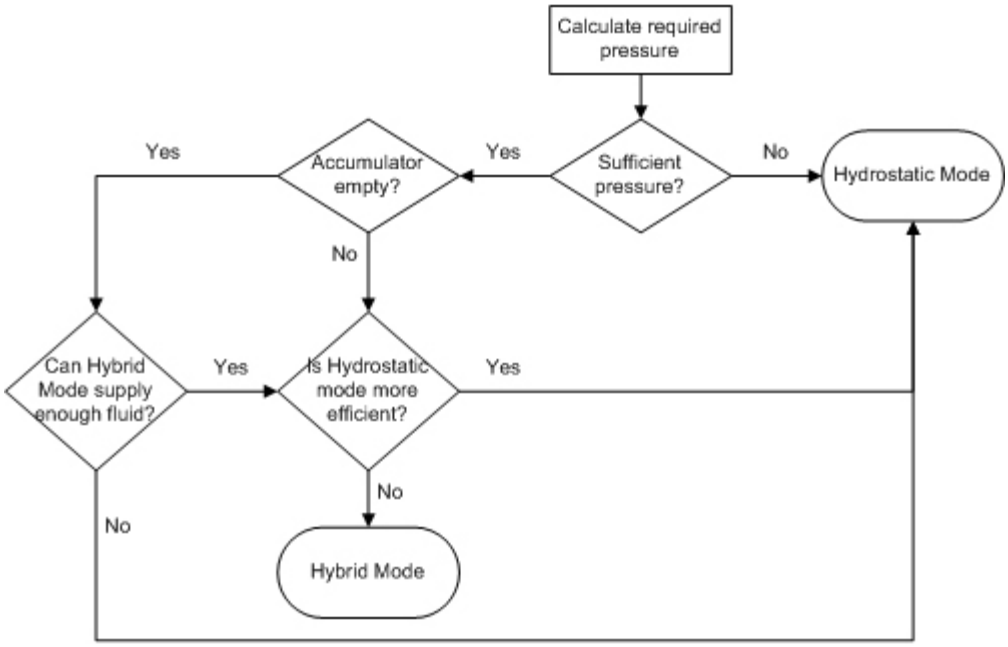


Figure 4.6: Strategy for hybrid/hydrostatic mode selection

4.3 Driving Cycles

Driving cycles have been developed for many different applications. The standard driving cycles used to test fuel economy in the United States for vehicles with a gross vehicle weight rating of less than 8500 lb_f are the Federal Test Procedure City and Highway driving cycles. Since these are the current standards, they were selected for use in the simulation. Velocity profiles for the city and highway cycles are shown in Figure 4.7 and Figure 4.8. Fuel mileage is also reported as a combined mileage which is 55% of the city mileage plus 45% of the highway mileage.

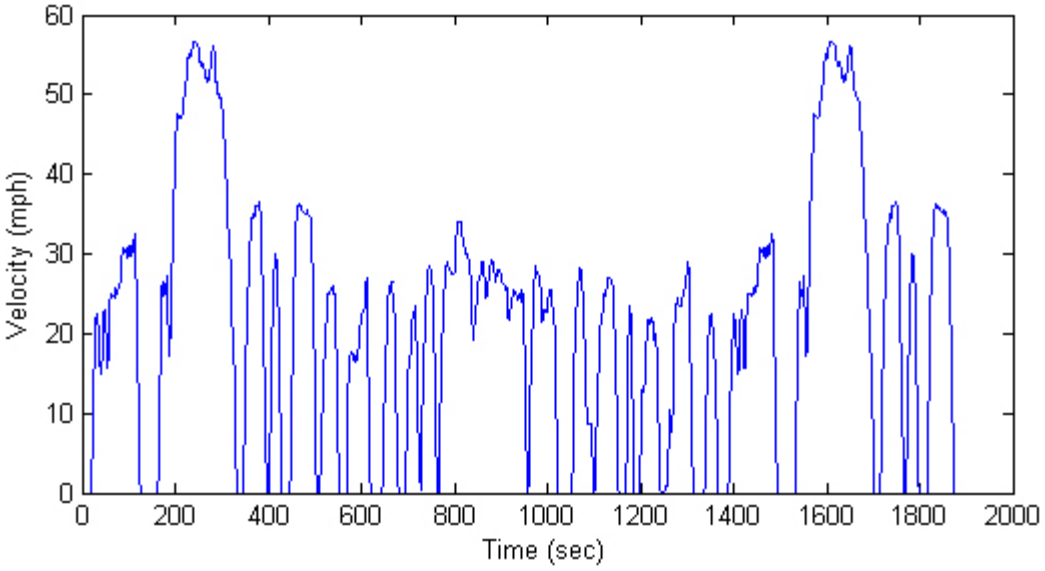


Figure 4.7: Federal Test Procedure City driving cycle

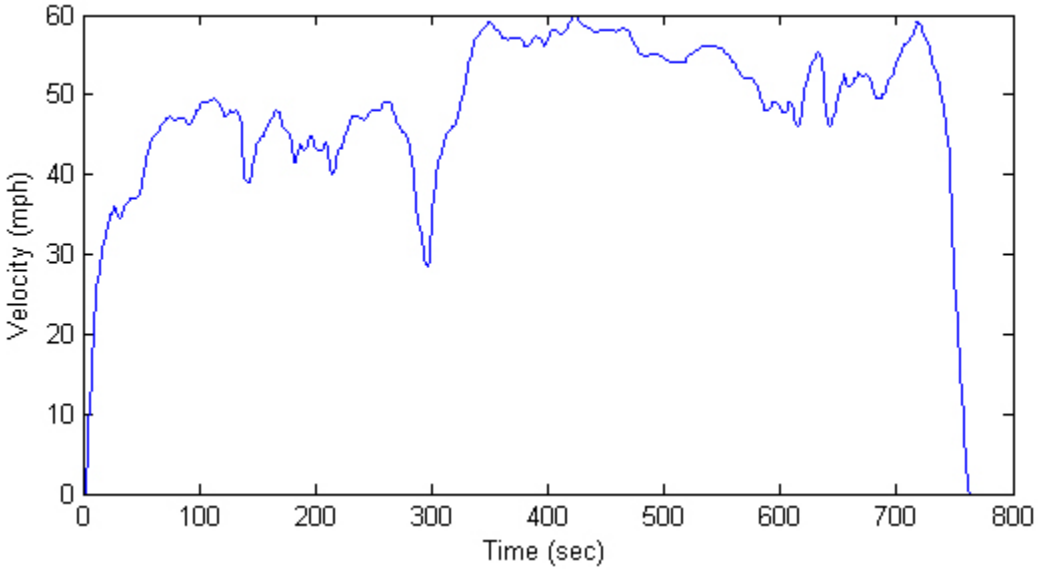


Figure 4.8: Federal Test Procedure Highway driving cycle

Chapter 5: **Results and Discussion**

5.1 Convergence

Convergence tests were performed using the city, highway, and SC03 driving cycles to determine an appropriate time step to use for the simulations. While the SC03 driving cycle is not used for the rest of the simulations, it was added here as an extra check for convergence. The results of the convergence test to determine an appropriate time step are shown in Figure 5.1. A time step of 0.05 seconds was chosen for the simulations based on these results.

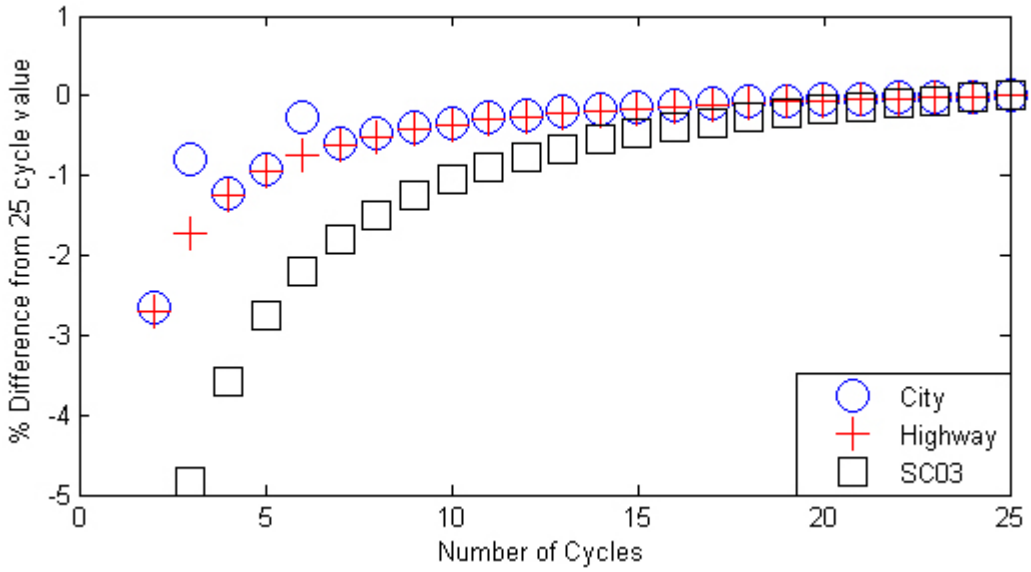


Figure 5.1: Convergence test results for determining time step

Since the stored energy in the initial and final states of the vehicle are generally not the same for a driving cycle, the reported fuel mileage may be inaccurate if the difference in stored energy is significant in terms of the energy used in the cycle. Running a driving cycles multiple times can help mitigate the effect of the stored energy difference. Figure 5.2 shows the results of running the driving cycles 25 times. 10 cycles was determined to be a sufficient number of cycles to provide acceptable results. After 10 cycles, the city and highway driving cycles were within 0.5% of their values at 25 cycles. The SC03 cycle value changed by 1% between 10 and 25 cycles.

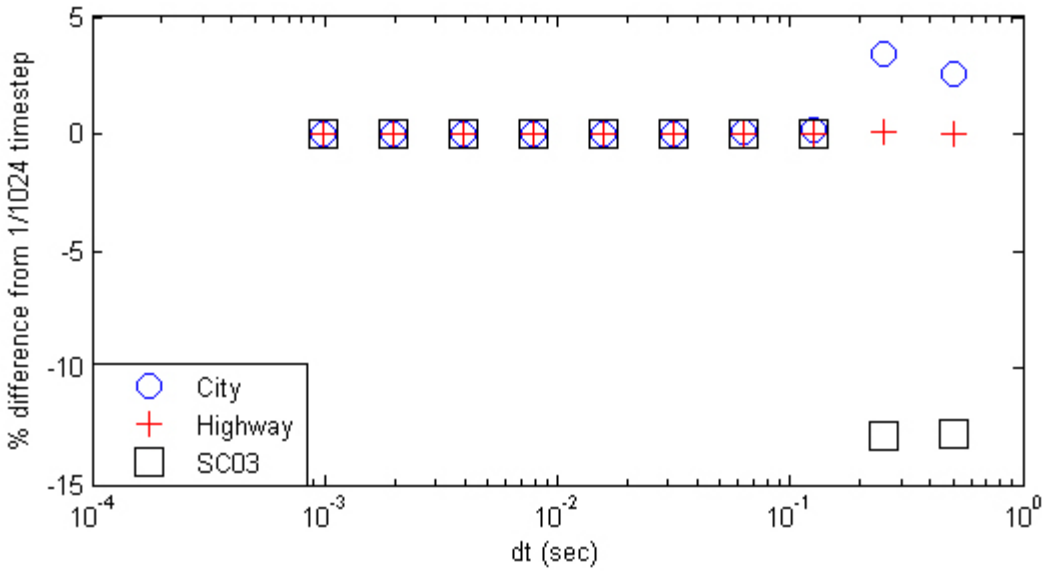


Figure 5.2: Convergence test results for determining number of cycles

5.2 HVDT Vehicle

The HVDT vehicle simulation was run through the city and highway cycles using charge pressures of 3000, 4000, and 5000psi. Simulation parameters are shown in Figure 5.1.

Table 5.1: HVDT simulation parameters

Parameter	Value	Units
Weight	6300	lb_f
Pump Displacement	180	cc
Motor Displacement	250	cc
Precharge	1000	psi
Buffer	100	psi
dt	0.05	sec

5.2.1 Overall Results

Figure 5.3 shows the mileage for both driving cycles and the combined mileage for the HVDT vehicle at the different charge pressures. Mileage was highest in the city cycle with a charge pressure of 3000 psi at 34.3 mpg. Increasing the charge pressure to 4000 psi and 5000 psi caused approximately 3% and 11% drops in mileage respectively. Mileage for the highway cycle was also highest with a charge pressure of 3000 psi at 27.2 mpg. Charge pressures of 4000 psi and 5000 psi caused mileage decreases of 3% and 4% respectively.

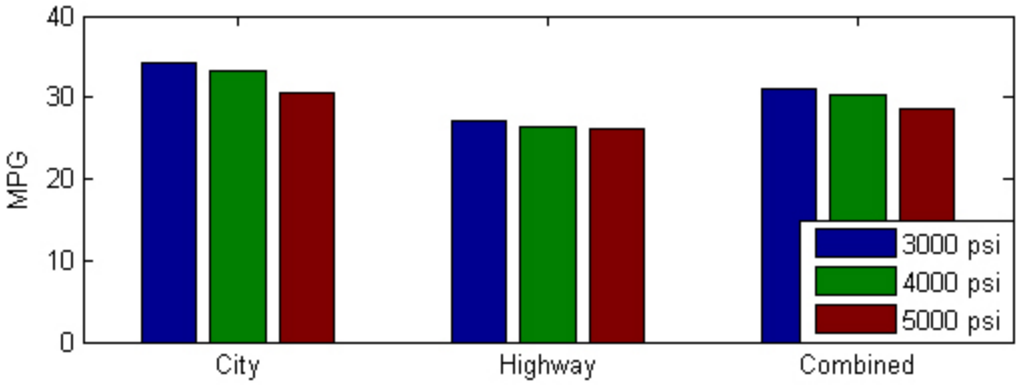


Figure 5.3: Mileage at different charge pressures for the HVDT vehicle

5.2.2 City Cycle

Energy use for the 10 city cycles is shown in Figure 5.4. The bars labeled acceleration represent the amount of energy required for the vehicle to follow the driving cycle with no losses. The bar labeled regen shows the total amount of energy reclaimed through regenerative braking. All of the other bars indicate the amount of energy lost due to their corresponding label.

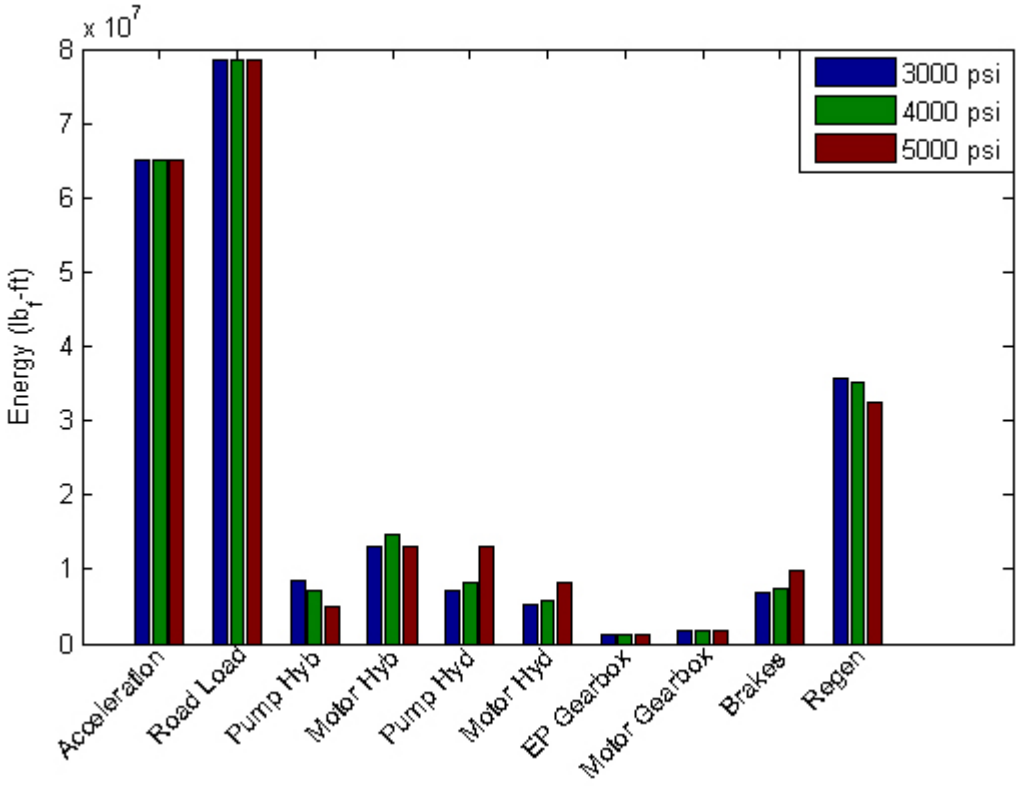


Figure 5.4: Energy use for 10 city driving cycles for the HVDT vehicle

The amount of energy lost through pump and motor inefficiencies in hydrostatic mode corresponds to the amount of time spent in hydrostatic mode shown in Figure 5.5. This indicates that the operation of hydrostatic mode is not dependent on the charge pressure which is an expected result as the mode does not interface with the energy storage system.

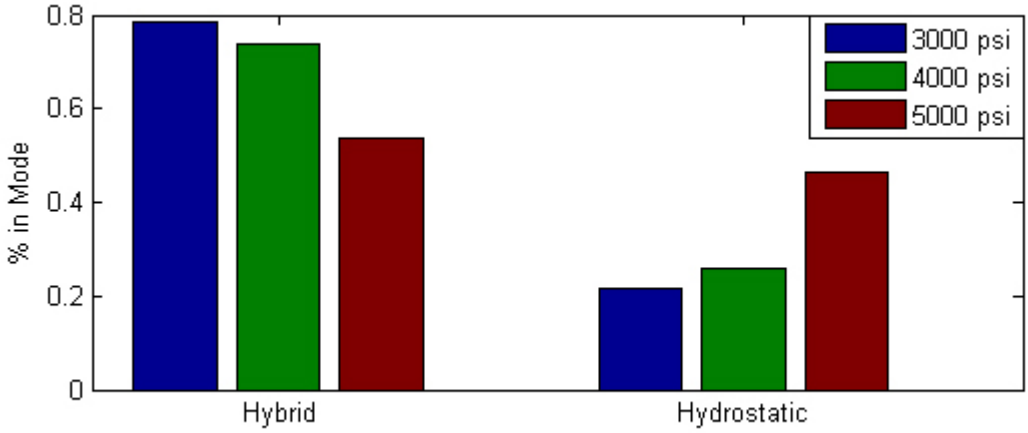


Figure 5.5: Percentage of time driving in operational mode during city cycles for the HVDT vehicle

This correlation also exists for the amount of energy lost through pump inefficiencies in hybrid mode. While hybrid mode interacts with the energy storage system, the pump efficiency stays within 5% of its maximum value from 1750 psi to 4930 psi (Figure 5.6) - about 80% of its operating range.

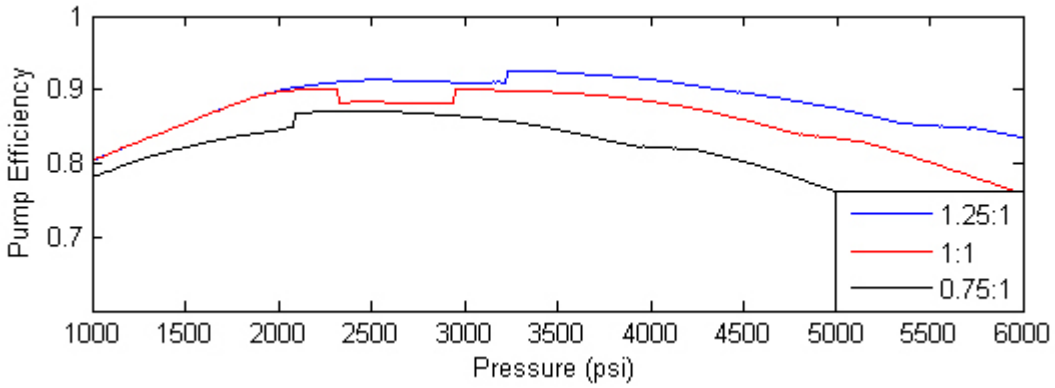


Figure 5.6: Pump efficiency in hybrid mode for different gear ratios

Unlike the pump, energy loss through the motor during hybrid mode does not correlate to the amount of time spent in the mode. The amount of energy lost with a charge pressure of 4000 psi is 13% greater than

at 3000 psi while the mode is used 6% less. At a charge pressure of 5000 psi, roughly the same amount of energy is lost through motor inefficiencies as at 3000 psi although 31% less time is spent in the mode. This indicates that, in this case, the motor is used much less effectively at higher pressures. The higher average system pressures, shown in Figure 5.7, cause a decrease in displacement which in turn causes a decrease in volumetric efficiency. Figure 5.8 and Figure 5.9 show these trends for a charge pressure of 5000 psi. The vertical axis shows the relative frequency of a particular data point in relation to the other data points on that graph. The z-scale is not the same for the two figures.

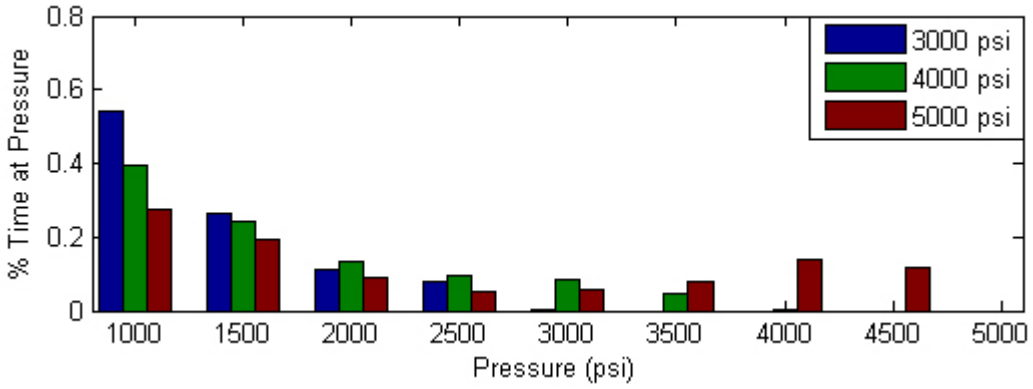


Figure 5.7: Pressure profile during city cycles for the HVDT vehicle

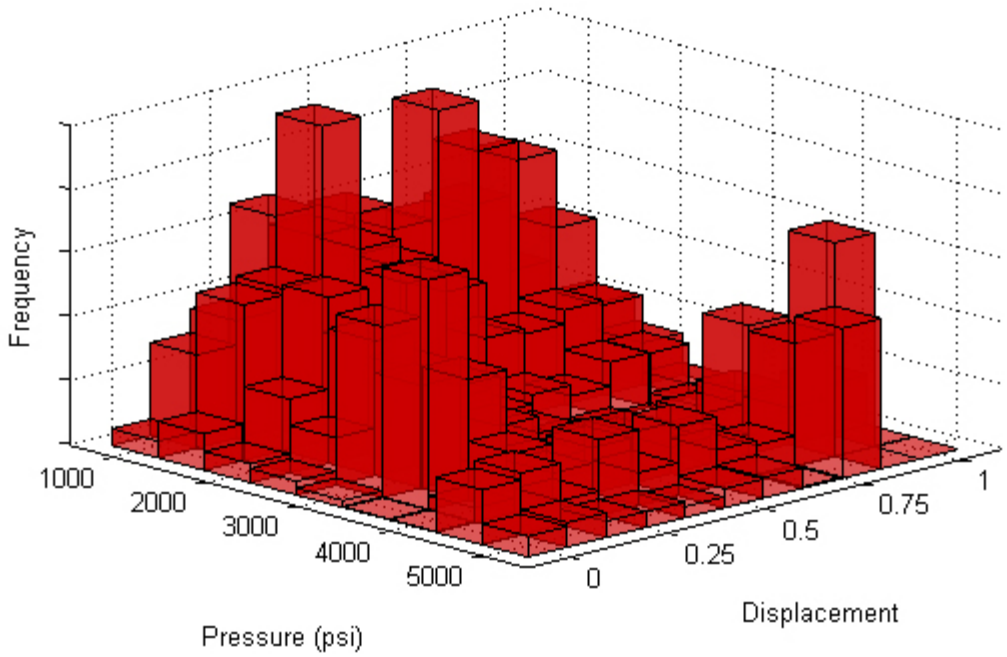


Figure 5.8: Motor displacement and pressure relationship in hybrid mode with 5000 psi charge pressure during city cycles for HVDT vehicle

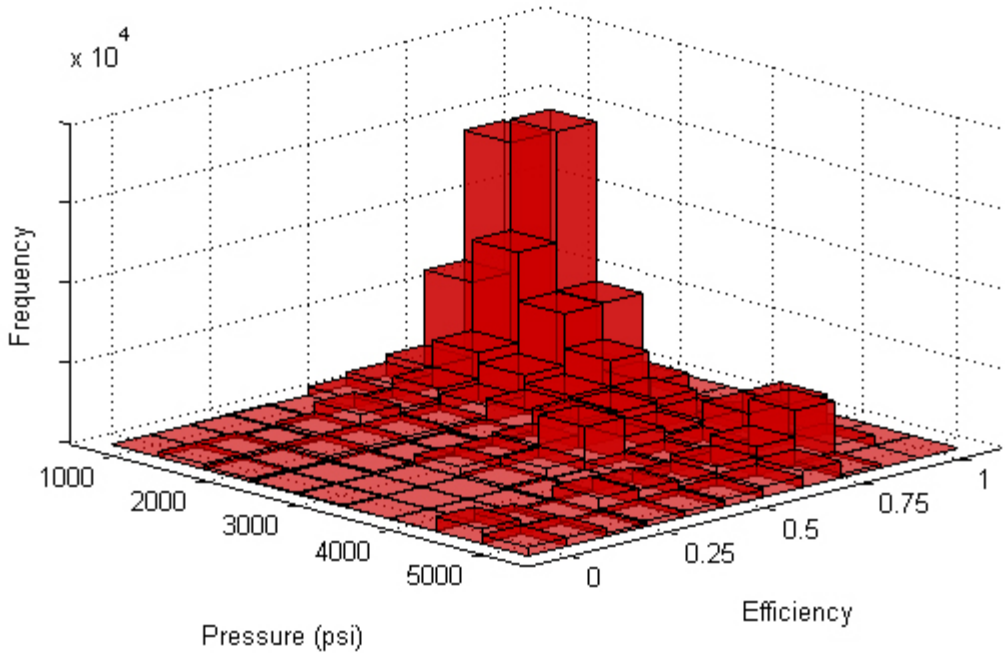


Figure 5.9: Motor Efficiency and pressure relationship in hybrid mode with 5000 psi charge pressure during city cycles for HVDT vehicle

Regenerative braking was also the most effective with the lowest charge pressure. With a charge pressure of 3000 psi, regenerative braking reclaimed 84% of the available energy while 77% was reclaimed with a charge pressure of 5000 psi. The difference in effectiveness is due to the decreased efficiency of the motor at higher pressures. Available energy is the vehicle's kinetic energy less the losses due to drag, rolling resistance, and gearbox efficiency.

Engine operation also varies significantly at the different charge pressures. Figure 5.10 and Figure 5.11 show the distribution of BSFC during 10 city cycles in the hydrostatic and hybrid modes respectively. Since the hybrid mode is not as efficient at the higher charge pressures, the best efficiency strategy shifts more of the driving cycle into the hydrostatic mode where the engine is not used as effectively at light loads. Figure 5.12 shows that the engine spends a large amount - 75% with a charge pressure of 3000 psi and 66% at 5000 psi - of the driving cycle shut off during the city driving cycle which helps conserve fuel.

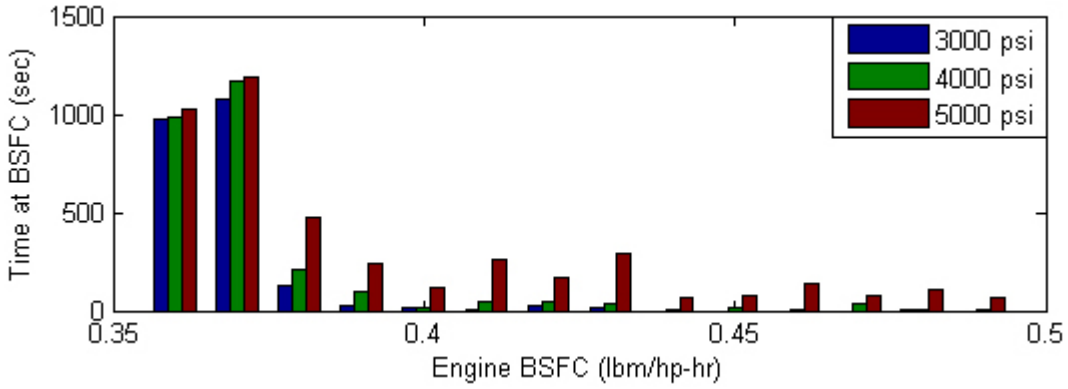


Figure 5.10: Engine BSFC profile in Hydrostatic mode for city cycles for HVDT vehicle

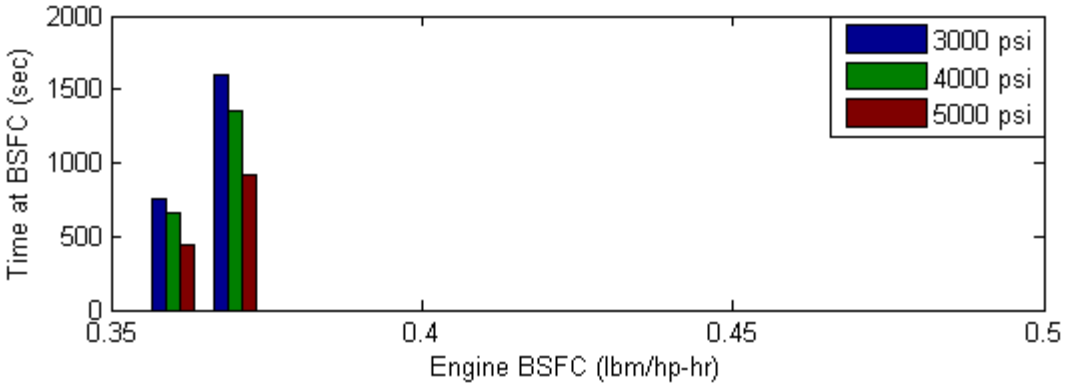


Figure 5.11: Engine BSFC profile in Hybrid mode for city cycles for HVDT vehicle

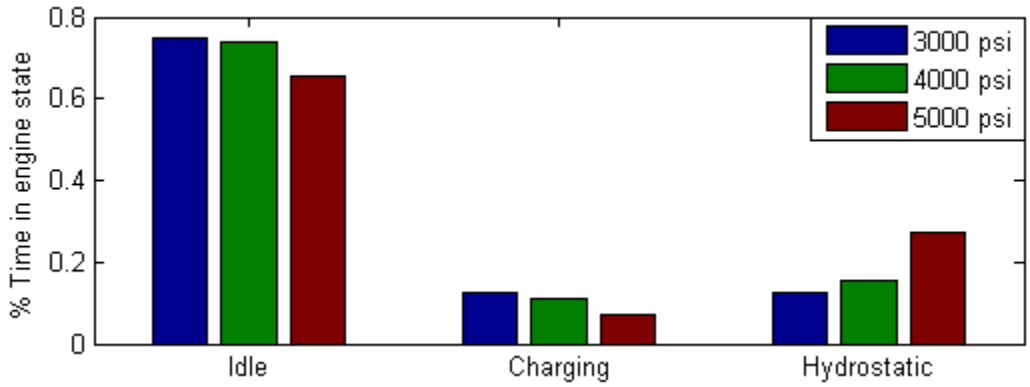


Figure 5.12: Engine state distribution for HVDT vehicle

To summarize, for this vehicle configuration, higher charge pressures for the city cycle cause a decrease in mileage. This is due to a decrease in motor displacements while in hybrid mode which lead to lower efficiencies. The reduced motor efficiency also decreases the effectiveness of regenerative braking causing a further decrease in fuel efficiency. The efficiency decrease of hybrid mode combined with the best efficiency strategy shifts more of the driving cycle into the hydrostatic mode where the engine is not used effectively at light loads.

5.2.3 Highway Cycle

Energy use for the highway cycle only varies by a few percent for the different charge pressures as shown in Figure 5.13. This can be explained by Figure 5.14 which shows that the vast majority of the highway driving cycle is spent in the hydrostatic mode whose efficiency independent of system pressure.

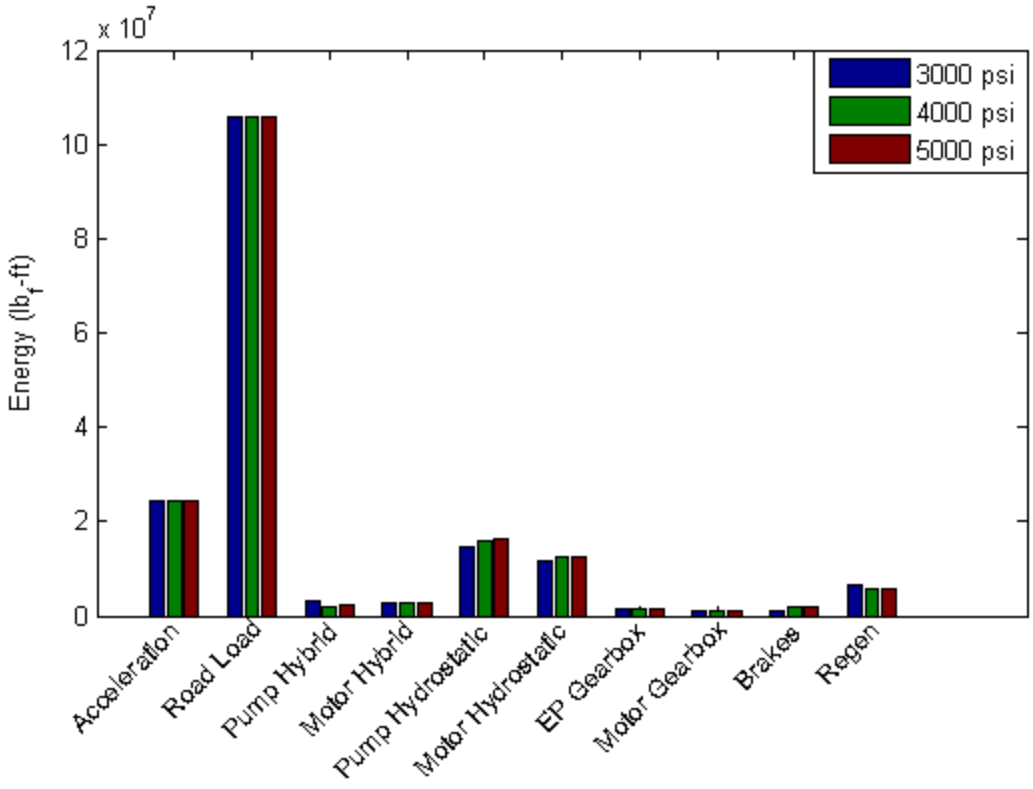


Figure 5.13: Energy use for 10 highway cycles for the HVDT vehicle

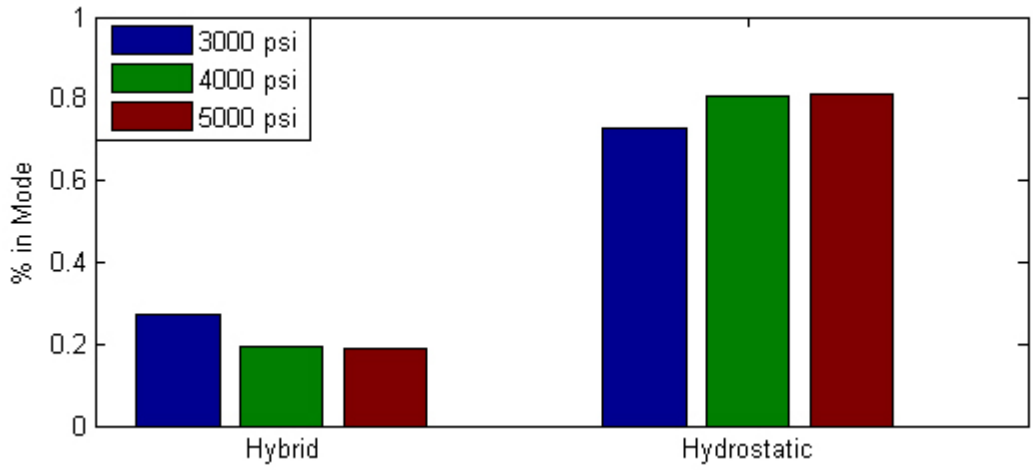


Figure 5.14: Percentage of time spent driving in operational mode during highway cycles for HVDT vehicle

The operation of the hydrostatic mode in the highway cycle is largely unaffected by charge pressure due to spending less than a third of the driving cycle in hybrid mode. Figure 5.15 shows that at a charge pressure of 5000 psi about 14% of the time spent in hybrid mode are at pressures significantly higher than

with a charge pressure of 3000 psi. When considering that this is 14% of approximately 20% of the driving cycle, this makes up a very small portion of the driving cycle.

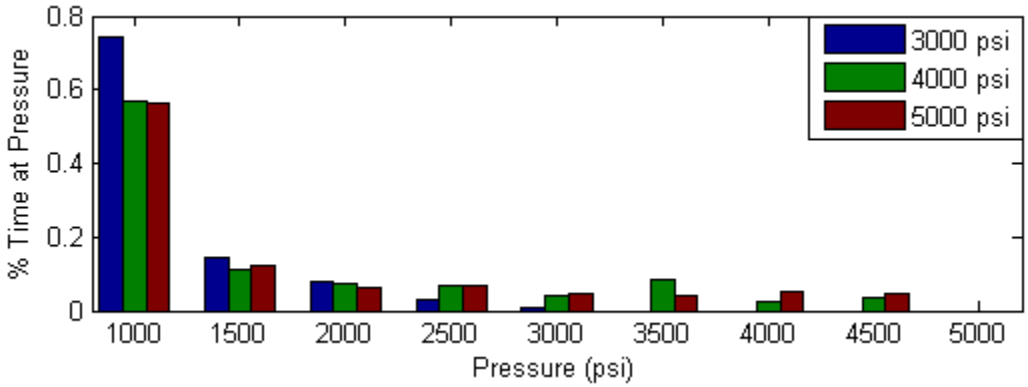


Figure 5.15: Pressure distribution during highway cycles for HVDT vehicle

Hybrid mode's reduced usage in the highway cycle, compared to the city driving cycle, is due to the higher power requirements. When the power requirements are low the engine is normally lightly loaded which is inefficient. Hybrid mode shifts the operating point of the engine in this situation by increasing the load with the balance of the energy being stored. Figure 5.16 shows the most efficient power output from the pump for across its pressure range. Once the power demand from the vehicle reaches these curves, increasing the load on the engine causes a drop in overall efficiency as the pressure rises. The decrease in overall efficiency is caused by the reduced motor displacement at pressures higher than optimal. Since hydrostatic mode does not interface with the energy storage system, it avoids non-optimal pressures. The power distribution for the highway cycle is shown in Figure 5.17.

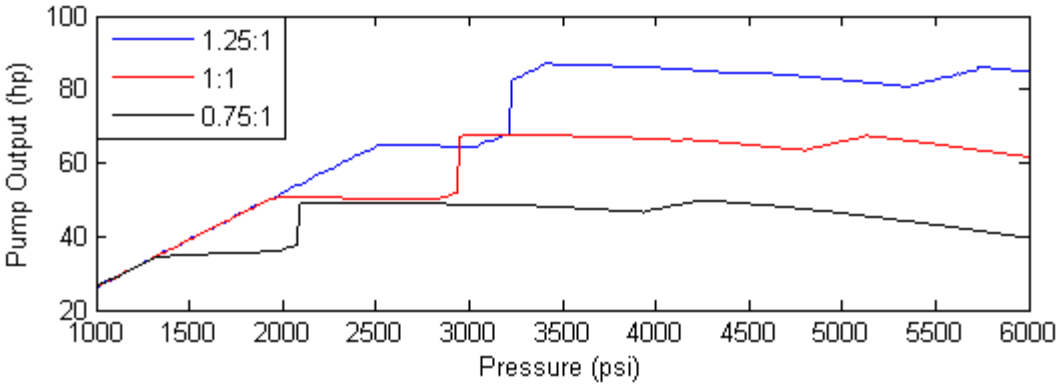


Figure 5.16: Hybrid mode pump power output for HVDT vehicle

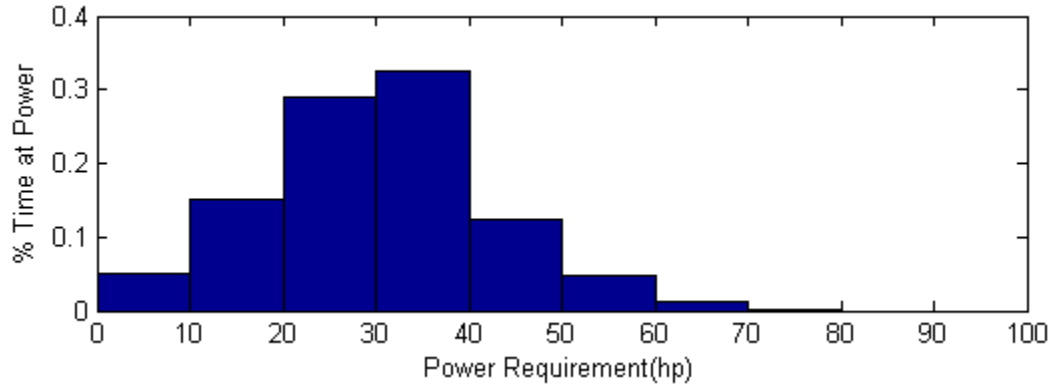


Figure 5.17: Power requirement distribution during highway cycles for HVDT vehicle

The vehicle comes to a stop once during the highway driving cycle and as such regenerative braking has little effect on the fuel efficiency for this driving cycle. It does, however, reclaim approximately 84% of the available energy for with a charge pressure of 3000 psi and 73% at 5000 psi.

5.3 Assumptions and Limitations

5.3.1 Valve Pressure Drops

The valves chosen for the hydraulic system were selected to keep pressure drops to a minimum at the system's peak flow rates. Pressure drops across the valves were considered to be small and were neglected in the simulation to simplify the model. Figure 5.18 shows the pressure drop distribution for a single valve in the h-bridge manifold during a combined city and highway driving cycle using data provided in its datasheet.(Parker Hannifin Corporation, 2008b) The pressure loss due to the valves in the h-bridge, one high side and one low side, is less than 6 psi for 95% of the simulation. At the lowest system pressure, 1000 psi, this is equates to a 0.6% loss. This assumption seems reasonable.

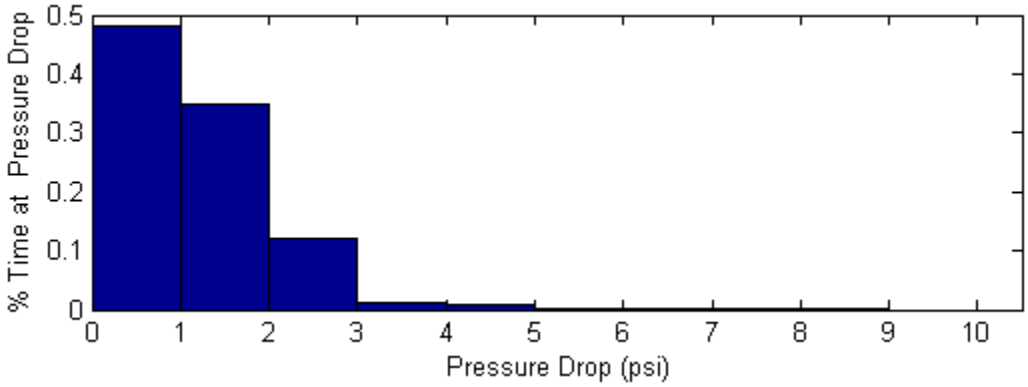


Figure 5.18: Pressure drop distribution for combined cycle for HVDT vehicle

The pressure drop across the post-pump check valve and filter were also neglected. Figure 5.19 shows the combined pressure drop distribution for these components using data provided in their datasheets (Parker Hannifin Corporation, 2006; Parker Hannifin Corporation, 2008a). The pressure drop due to these components is at best a 1% loss and at worst a 6% loss. Including the check valve and filter efficiencies would improve the accuracy of the simulation.

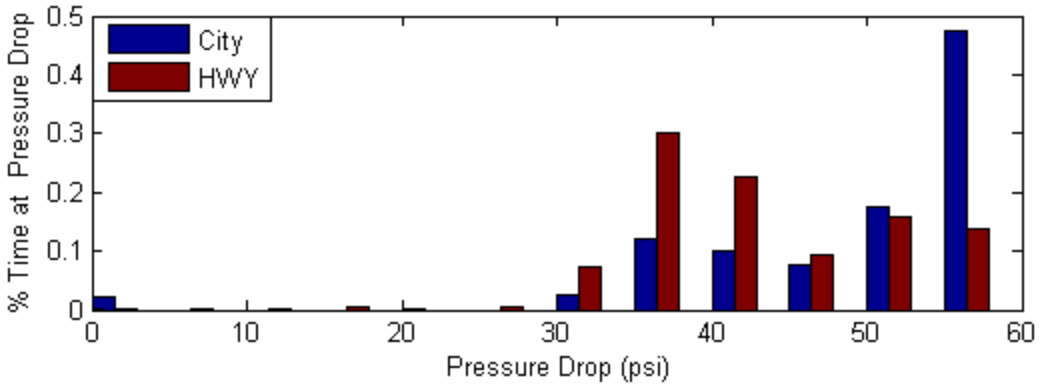


Figure 5.19: Check valve and filter pressure drop distribution for HVDT vehicle

5.3.2 Motor and Pump Performance

Efficiency data for the motor and pump at less than full displacement was not available. Off-peak performance was modeled by assuming a constant leakage flow with respect to displacement. This assumption predicts efficiencies that are lower than the actual values and becomes less accurate as displacement decreases. Figure 5.20 shows the displacement distribution for the motor and pump in both driving cycles while in hydrostatic mode. The pump operates almost exclusively above 60% displacement in hydrostatic mode which is above where the model's accuracy begins to decline rapidly. The motor generally runs at displacements greater than 40%. This is above where the model's predicted efficiencies begin to drop of quickly. When spot checking the data and comparing it to the maps in the EPA report, the model's efficiencies are low but within 5-10%. However, the efficiency maps in the EPA report are for components that are more efficient at full displacement as well. The assumption used seems to be reasonable for use with hydrostatic mode.

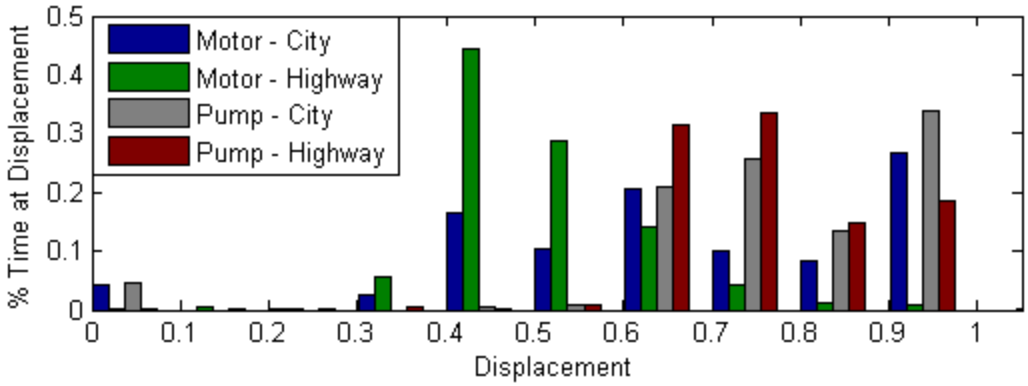


Figure 5.20: Displacement distribution in hydrostatic mode for HVDT vehicle

Figure 5.21 shows the displacement distribution of the motor and pump during the city and highway driving cycles while in hybrid mode for the HVDT vehicle. The pump operates at nearly full displacement while in hybrid mode. The motor, however, has a fairly even distribution across the displacement range. This becomes problematic below 40% displacement as the volumetric efficiency begins to fall off rapidly. Figure 5.22 shows the distribution of energy loss for the motor in hybrid mode for ten sets of both driving cycles for the HVDT vehicle. The losses here correspond to those in Figure 5.4. Approximately half of the energy loss occurs at displacements of 40% and less. This is a trivial amount for the highway driving cycles but is approximately 4% of the losses in the city driving cycles. The efficiency model is reasonable for the pump but efficiency data for the motor would be beneficial.

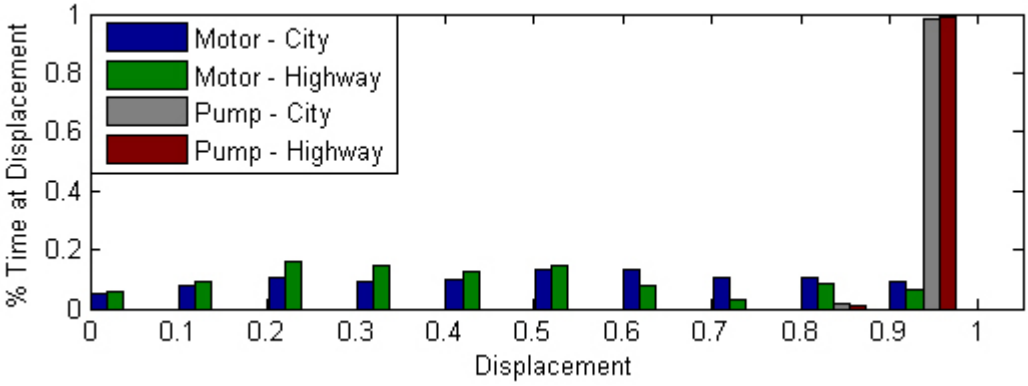


Figure 5.21: Displacement distribution in hybrid mode for HVDT vehicle

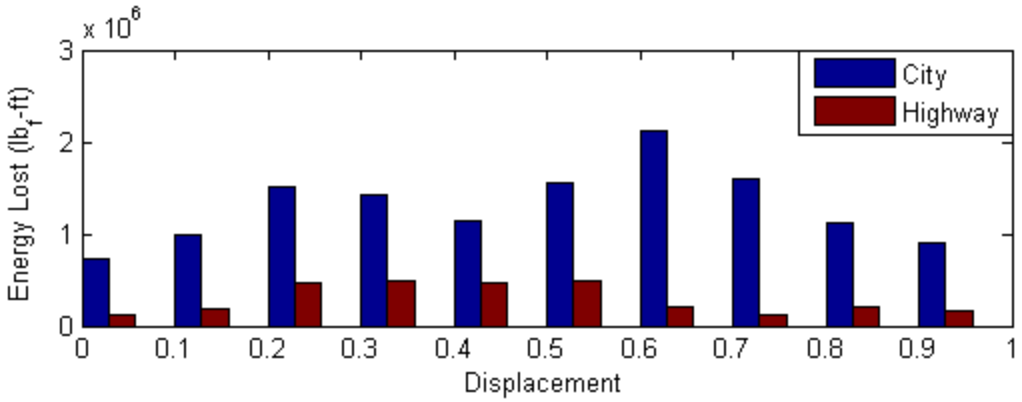


Figure 5.22: Distribution of energy loss for the motor in hybrid mode for HVDT vehicle

Motor efficiency data was also not available at speeds below 260 rpm. Since the driving cycles include periodic stops, this efficiency data had to be extrapolated. Figure 5.23 and Figure 5.24 show the distribution of motor speeds in the hybrid, hydrostatic, and regen operational modes for the city and highway driving cycles. Hydrostatic mode is unaffected by the extrapolated data in both the city and highway driving cycles. While in hybrid mode, the motor uses extrapolated data approximately 10% of the time during the city driving cycle and 1% of the time during the highway cycle. Referring back to Figure 5.5 and Figure 5.14, this represents 8% of the time spent driving the vehicle during the city driving cycle and less than 1% during the highway driving cycle. Low speed data for the motor would improve the accuracy of the simulation for hybrid mode during the city driving cycles. The effect of low speed data on the hybrid mode during highway driving cycles would be negligible. Regenerative braking is most affected by the extrapolated data. This data is used for approximately 18% of regen mode during the city driving cycle and 10% during the highway driving cycle. Low speed data for the motor would be useful for the regenerative braking portions of the simulation.

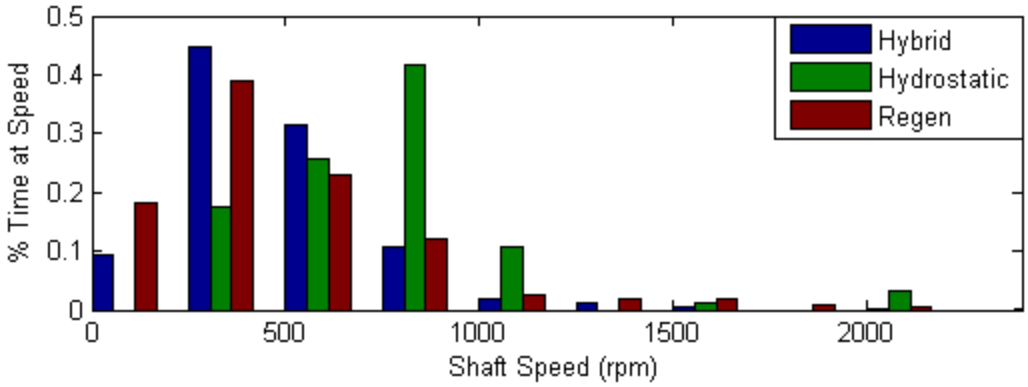


Figure 5.23: Motor speed distribution during city cycles for HVDT vehicle

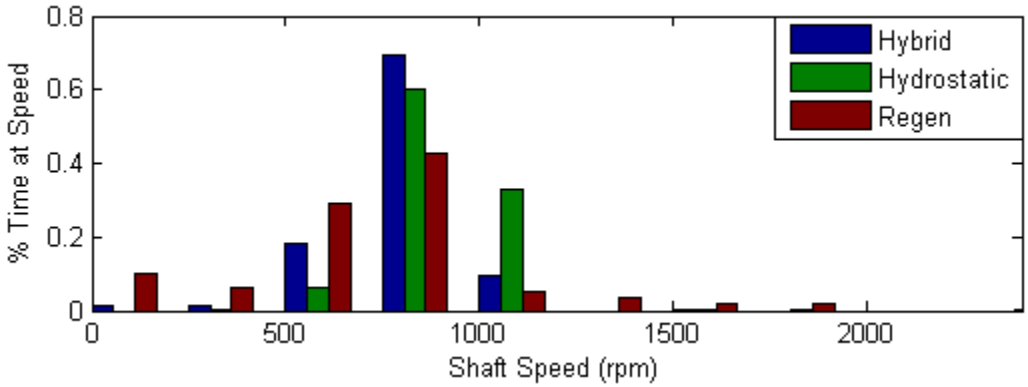


Figure 5.24: Motor speed distribution during highway cycles for HVDT vehicle

Low speed efficiency data for the pump was also unavailable. The data for the pump was much more limited than for the motor with efficiency data only going down to 1000 rpm. Since operation below this point was not required, data was not extrapolated. Figure 5.25 shows the distribution of pump speeds for hybrid and hydrostatic modes for both driving cycles. The pump operated almost exclusively at 1000 rpm in both modes and during both driving cycles. This strong preference for low speed operation makes low speed data desirable.

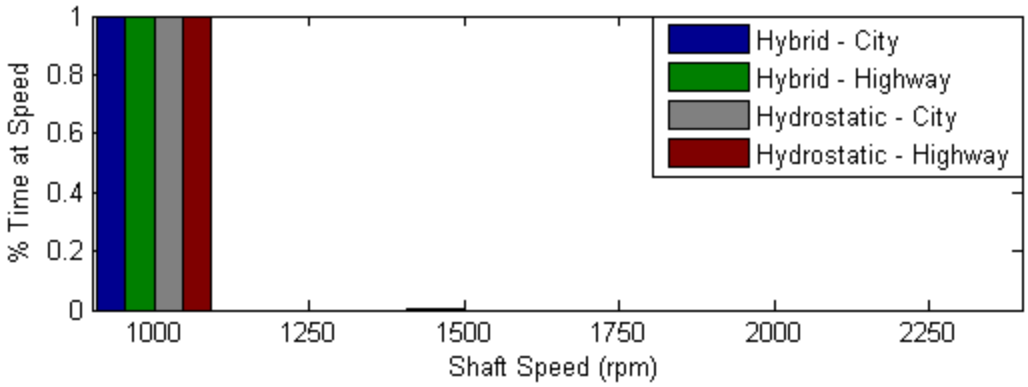


Figure 5.25: Pump speed distribution for HVDT vehicle

5.4 EPA Vehicle

The simulation was also run using a configuration similar to the one used in the EPA's simulations. This was done as a reference since there is no data from the HVDT vehicle to compare to. The EPA vehicle was run through the city and highway cycles using charge pressures of 3000, 4000, and 5000psi. Simulation parameters are shown in Table 5.2.

Table 5.2: EPA simulation parameters

Parameter	Value	Units
Weight	5750	lb_f
Pump Displacement	135	cc
Motor Displacement	135	cc
Precharge	1000	psi
Buffer	100	psi
dt	0.05	sec

Figure 5.26 shows the mileage for both driving cycles and the combined mileage for the HVDT vehicle at the different charge pressures. Mileage was nearly the same for all charge pressures at 36.7 mpg for the city driving cycle and 28.5 mpg for the highway driving cycle. This is slightly higher than the EPA estimates of 35.9 mpg and 28.2 mpg for the city and highway driving cycles respectively. These simulations predicted mileage is approximately 2% and 1% high for the city and highway cycles respectively. However, the difference between the Perkins and EPA engine efficiencies, approximately

10%, must be taken into account. The peak thermal efficiency of the engine the EPA used in their simulations was 41.9% while the peak thermal efficiency of the Perkins was 37.9%. The efficiency maps the EPA used for their simulation indicate that the pump and motor they simulated are more efficient that the units used in this simulation. The EPA motor and pump are roughly 2-4% more efficient. The efficiency model used for the motor and pump introduces an additional error of up to 7% at 50% displacement. Figure 5.27 shows the pressure drop due to the check valve and filter at the output of the pump. This represents a 2-4% loss that is not accounted for in the reported fuel mileage. The accumulator efficiency was also neglected in this simulation which is another 4% loss without including thermal losses (Pourmovahed, Beachley, & Fronczak, Modeling of a Hydraulic Energy Regeneration System - Part I: Analytical Treatment, 1992a). When combined, these efficiencies would increase the reported fuel mileage by about 6% for the city driving cycle and 7% for the highway driving cycle. This places the simulation's calculated mileage about 8% higher than the EPA's estimates. The 8% difference is before thermal losses and operation of the kidney loop are taken into account which would lessen the difference.

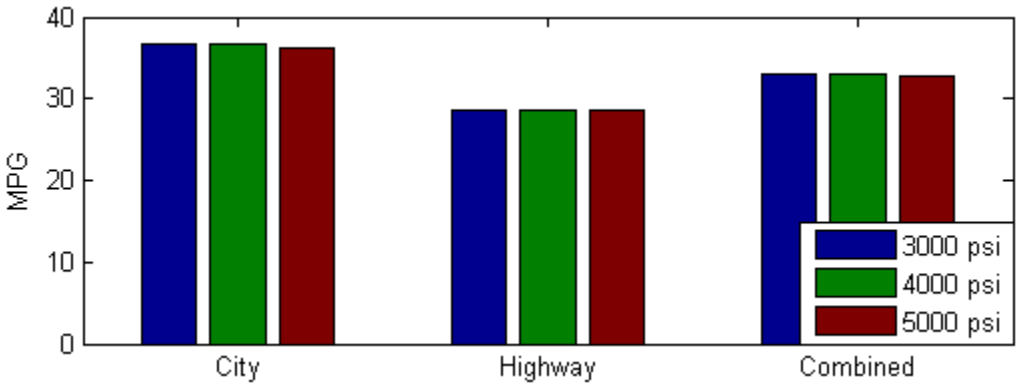


Figure 5.26: Mileage at different charge pressures for the EPA vehicle

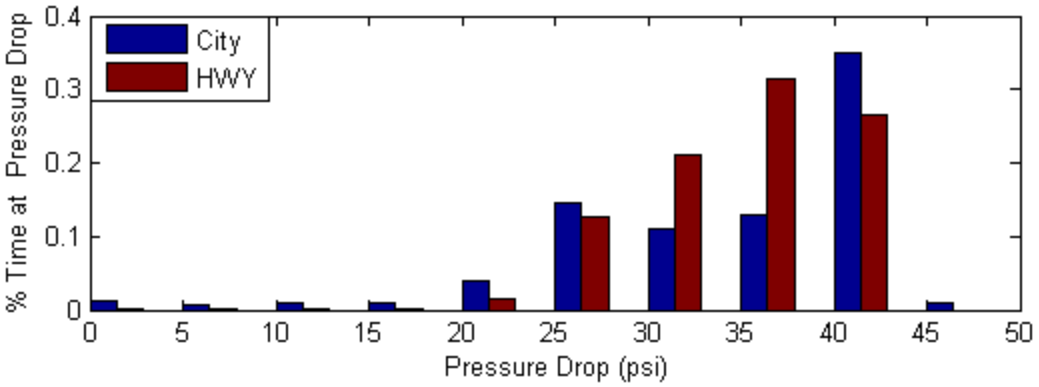


Figure 5.27: Check valve and filter pressure drop distribution for EPA vehicle

Chapter 6: **Conclusion and Future Work**

This thesis aimed to create an accurate simulation of a hydraulic series hybrid vehicle which could be used as a design and development tool. The results from the simulation are fairly accurate when compared to the EPA's estimates. The accuracy of the simulation could be enhanced by accounting for the pressure drops due to the post-pump filter, hoses, valves, and manifolds. Data for the filter, hoses, and valves are commonly available in their data sheets but this is not the case for manifolds. The pressure drop due to the h-bridge valves should be considered even though they were very small in this case so smaller valves can be simulated without reducing accuracy. Due to the high cost of manifolds, measurement of pressure drops for a range of configurations is not feasible and estimates would need to be made. The accumulator's frictional losses should also be taken into account as approximately 4% of the stored energy is lost. The accumulator's thermal losses should be investigated to determine their signifigance. Pourmovahed et al. indicate that the open-cell foam significantly reduces thermal losses; however, their total accumulator volume was ten times smaller than HVDT's configuration. The kidney-loop's energy consumption must be analyzed which will most likely require bench testing of the system. Since it is not practical to run tests in order to collect performance data at less than full displacement for many motors and pumps and manufacturers do not readily make this data avaialble, assumptions regarding these efficiencies will have to be made when using the simulation as a design tool. When using the simulation as a development tool, the performance of the selected motors and pumps should be measured.

The model and simulation are highly configurable with options to enable and disable operational modes, a configuration file that allows changing of all vehicle parameters, and the ability to accept arbitrary driving cycles and performance maps. Parallel processing is employed to minimize the amount of time a set of simulations consumes which makes it more useful for both design and development. This is useful when attempting to optimize aspects of the vehicle such as the motor displacement. Figure 5.17 shows the power requirements for the HVDT vehicle which can be used along with Figure 6.1 to determine if the motor is being used effectively. This shows that the motor is much too large and forced to operate within

a narrow range outside of its peak efficiency. The power axis on the map can be scaled to find a more appropriate motor displacement.

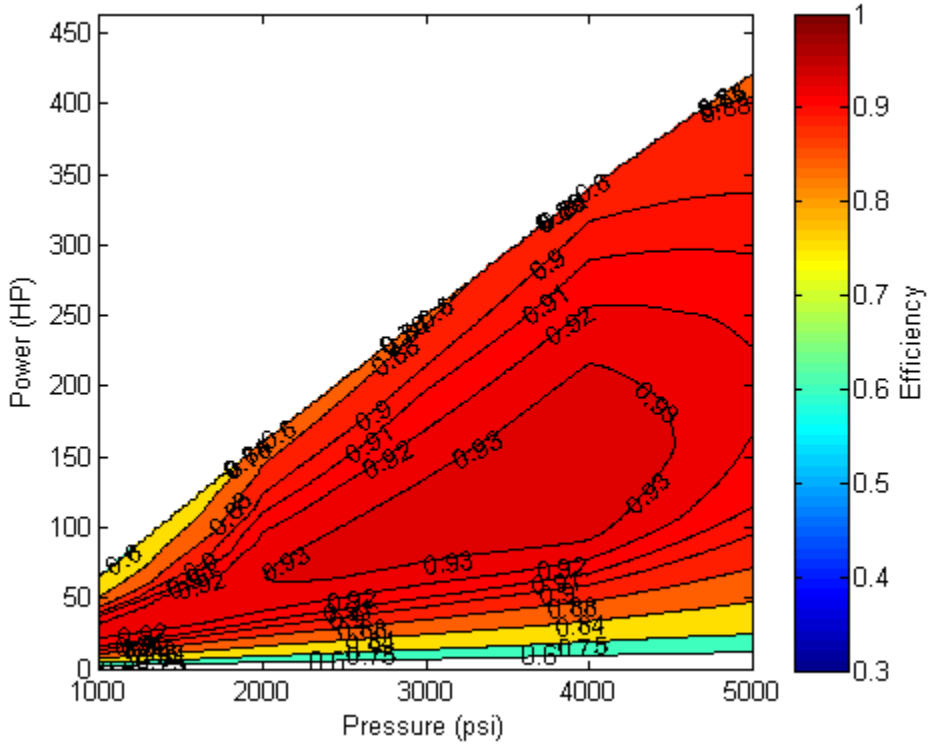


Figure 6.1: HVDT motor peak overall efficiency map

It is also possible to use the power requirements to determine if the motor and pump are matched well. Figure 6.2 shows the overall efficiency map for the HVDT motor at an output of 35 HP. This shows that the motor is most efficienct between 1100 and 1700 psi for the typical power requirements of the highway driving cycle. Figure 6.3 shows the overall efficiency map for the HVDT pump at an output of 38 HP. The pump is most efficient for this output between 1700 and 2900 psi. This shows that the motor and pump are not a good match. Efficiency data at higher displacements and pressures is not available for the pump due to the lack of low speed data.

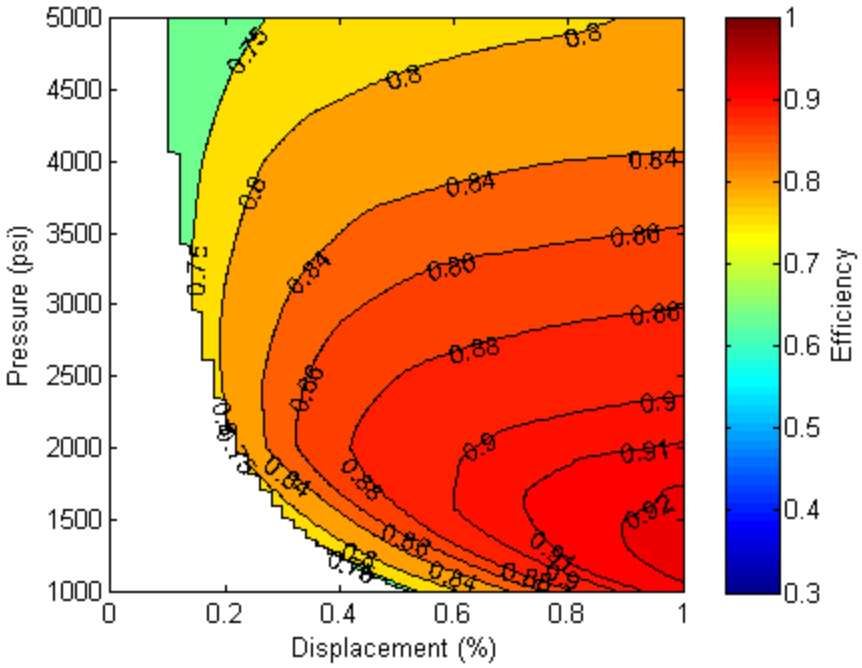


Figure 6.2: HVDT Motor peak efficiency map at 35 HP

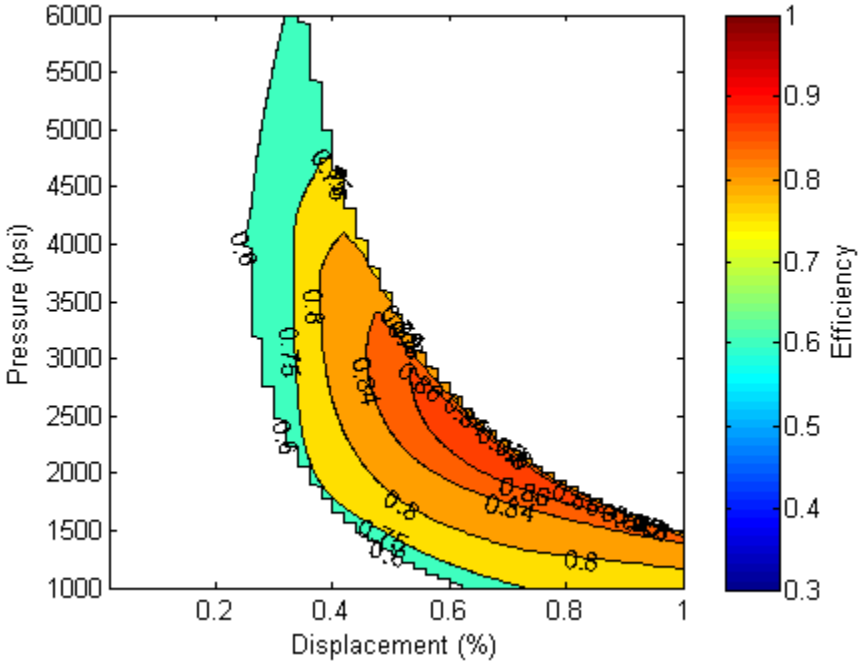


Figure 6.3: HVDT Pump peak efficiency map at 38 HP

The control strategy implemented in the simulation works well for a simulation where factors such as component transients and the driver are neglected. Switching between operating modes requires the motor and pump to rapidly change displacements and valves to open and close which can cause spikes in pressure and torque. This needs to be addressed in the hydraulic system. Frequent on-off cycling of the engine may not be acceptable to a driver and limits on the frequency of cycling may need to be implemented. Another alternative would be to set the system to charge when switched out of hydrostatic mode. A gap exists between hybrid mode not being able to supply enough fluid and running hydrostatic mode where running hybrid mode in a slightly less efficient configuration may be more efficient than hydrostatic mode. It would be beneficial to either extend hybrid mode or create a new mode to fill this gap. The simulation could also be improved by making the operational modes and control strategy completely modular such that the operational modes could be used within any control strategy and control strategies could be selected on the fly depending on the driving cycle.

Bibliography

Alson, J., Barba, D., Bryson, J., Doorlag, M., Haugen, D., Kargul, J., et al. (2004). *Progress Report on Clean and Efficient Automotive Technologies Under Development at EPA.* Advanced Technology Division Office of Transporation and Air Quality U.S. Environmental Protection Agency.

Buchwald, P., Christensen, G., Larsen, H., & S, P. P. (1979). *Improvement of Citybus Fuel Economy Using a Hydraulic Hybrid Propulsion System - A Theoretical and Experimental Study.* Warrendale: SAE Paper 790305.

Denison Hydraulics. (2003). *Gold Cup Series Piston Pumps for Open & Closed Circuits.*

Jacobsen, R. T., Stewart, R. B., & Jahangiri, M. (1986). Thermodynamic Properties of Nitrogen from the Freezing Line to 2000 K at Pressures to 1000 MPa. *Journal of Physical and Chemical Reference Data , 15* (2), 735-908.

Little, W. J., & A, N. C. (1962). *Tables of the Thermodynamic Properties of Nitrogen From 100 to 1500°K.* Arnold AEDC-TDR-62-170.

Manring, N. D. (2005). *Hydraulic Control Systems.* Hoboken, New Jersey: John Wuley & Sons, Inc.

Martins, R., Amaro, R., & Seabra, J. (2008). Influence of Low Friction Coatings on the Scuffing Load Capacity and Efficiency of Gears. *Tribology International* , 234-243.

Parker Hannifin Corporation. (2008a). *Flow, Check, Pressure Control, and Sandwich Valves.*

Parker Hannifin Corporation. (2006). *Hydraulic and Lube Filtration Products.*

Parker Hannifin Corporation Hydraulic Filter Division. (2005). *Handbook of Hydraulic Filtration.* Metamora.

Parker Hannifin Corporation. (2008b). *Industrial, Electrohydraulic and DIN Slip-in Cartridge Valves.*

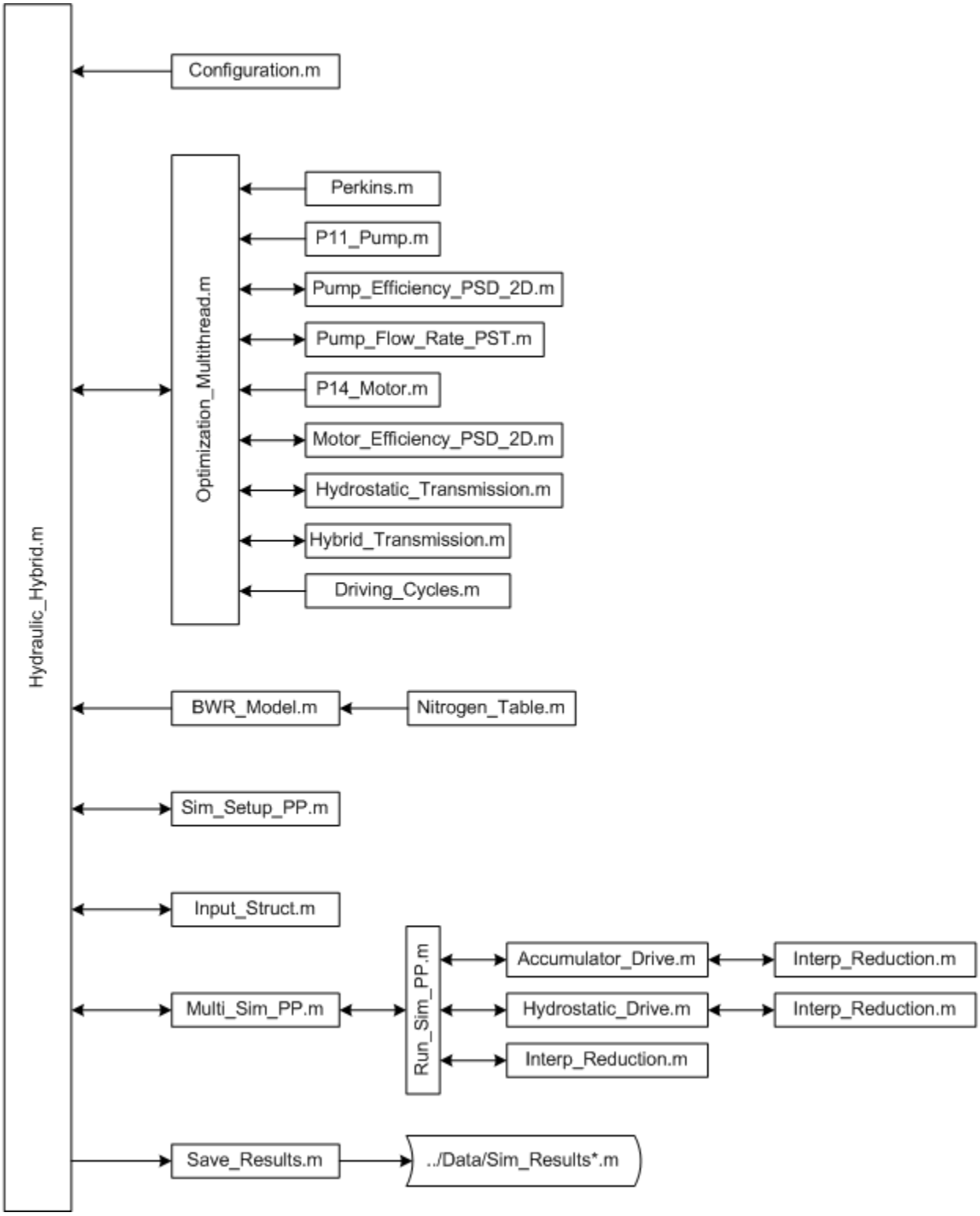
Pourmovahed, A. (1993). Sizing Energy Storage Units for Hydraulic Hybrid Vehicle Applications. *Advanced Automotive Technologies , 52*, 231-246.

Pourmovahed, A., Beachley, N. H., & Fronczak, F. J. (1992a). Modeling of a Hydraulic Energy Regeneration System - Part I: Analytical Treatment. *Journal of Dynamic Systems, Measurement, and Control* , 155-159.

Pourmovahed, A., Beachley, N. H., & Fronczak, F. J. (1992b). Modeling of a Hydraulic Energy Regeneration System - Part II: Experimental Program. *Journal of Dynamic Systems, Measurement, and Control* , 160-165.

Appendix A: Code Diagram

This diagram shows the relationship between the files and the flow of data.



Appendix B: Configuration.m

```
%% Configuration.m
% This file contains the model and system parameters
% Files must be in a folder named Simulation

%profile on -detail 'builtin'
%% Model_Configuration
Enable_Parallel_Processing = true;
Cores = 2;
addpath(genpath('..\Simulation'))

Simulation.continue = false;

Simulation.Interp2_Range = 15;
Simulation.Interp3_Range = 15;
Map_Steps = 19;

%% Environment
Environment.Air_Density = 0.00237;  %slug/ft^3
Environment.Diesel_Density = 7.09;  %lbm/gal
%% Hydraulics
%Hydraulics.Precharge = 2500;
Hydraulics.HP_ACC_Vol_Max = 9240; %in^3
Hydraulics.HP_ACC_Gas_Vol_Max = 0.03436*4; %m^3

%% Accumulator Configuration
Hydraulics.Precharge_Temp = 290; %K
Hydraulics.Nitrogen_Temp_init = 290; %K
Foam.mass = 1.496/0.015273*Hydraulics.HP_ACC_Gas_Vol_Max;
Foam.c = 2300;    % J/kg K

%% Vehicle Configuration
Vehicle.RL.A = 56.69;
Vehicle.RL.B = 1.1514; %0.78504545;
Vehicle.RL.C = 0.03121; %0.01450878;

Vehicle.Differential_Ratio = 3.73; %0.625
Vehicle.Tire_Diameter = 1.458*2; %ft
Vehicle.Weight = 6300; %lbf
Vehicle.CdA = 12.126; %ft^2
Vehicle.Crr = 0.03; %asphalt
Vehicle.Gearbox_Efficiency = 0.99;
Vehicle.Wheel_MOI = 0*4*3/4*(50/32.2)*(Vehicle.Tire_Diameter/2)^2;

if (isempty(DataFile))
Vehicle.EP_Gear_Ratios = [1.25 1 .75];
Vehicle.Motor_Gear_Ratios = [2 1 0.625 0.5];

%% Engine
Engine.Speed_Min = 1000;
Engine.Speed_Max = 2400;
Engine.Speed_Increment = 10;
Engine.Torque_Min = 1;
Engine.Torque_Max = 450;
Engine.Torque_Increment = 2;
Engine.Speed_Range = ...
```

```
        Engine.Speed_Min:Engine.Speed_Increment:Engine.Speed_Max;
Engine.Torque_Range = ...
        Engine.Torque_Min:Engine.Torque_Increment:Engine.Torque_Max;
Engine.Gear_Ratio = 1;
Engine.FFR_Idle = 1.1;

%% Motor
Motor.Pressure_Increment = 10;  %psi
Motor.Speed_Min = 0;
Motor.Speed_Max = 2400;
Motor.Speed_Range = Motor.Speed_Min:...
        (Motor.Speed_Max-Motor.Speed_Min)/200:Motor.Speed_Max;

%% Pump
Pump.Pressure_Increment = 10;   %psi
Pump.Speed_Range = Engine.Speed_Range/Engine.Gear_Ratio;
Pump.Torque_Range = 0:Engine.Torque_Increment: ...
        Engine.Torque_Max*max(Vehicle.EP_Gear_Ratios);

 %%  Simulation
Simulation.Energy_Pumped_Init = 0;
Simulation.BSFC_Energy_Total_Init = 0;
Simulation.Regen_Energy_Init = 0;


end
```

Appendix C: Optimization_Multithread.m

```
%% Optimization_Multithread.m
% This file loads the Engine, Pump, Motor, and driving cycle data. The file
% also calls the hybrid and hydrostatic optimization files.
%
%
% File Dependencies
%   -Hybrid_Transmission
%   -Hydrostatic_Transmission
%   -Pump_Efficiency_PSD_2D
%   -Pump_Flow_Rate_PST
%   -Motor_Efficiency_PSD_2D
%   -Driving Cycles

%% Initialize
display('Initializing');
warning off all

%% Retrieve Engine Data
display('Retrieving Engine Data');
[Engine.BSFC_Map] = Perkins(Engine.Speed_Range,Engine.Torque_Range);

%% Retrieve pump data
display('Retrieving Pump Data');
[Pump.Efficiency_Map ...
    Pump.Vol_Efficiency_Map ...
    Pump.Mech_Efficiency_Map ...
    Pump.Displacement_Map ...
    Pump.Pressure_Range ...
    Pump.Displacement_Steps ...
    Pump.Displacement_Max] ...
        = P11_Pump( ...
            Pump.Speed_Range, ...
            Pump.Torque_Range, ...
            Pump.Pressure_Increment);

%% Pump Calcs
Pump.Displacement_Range = ...
        (1:Pump.Displacement_Steps)/Pump.Displacement_Steps;

%An efficiency array with axes of pressure, speed, and torque is created
%for the pump
tic
Pump_Efficiency( ...
        1:size(Pump.Pressure_Range,1), ...
        1:size(Pump.Speed_Range,2), ...
        1:size(Pump.Torque_Range,2)) = [0];

display('Creating Pump Efficiency Array')
for index1 = 1:size(Pump.Pressure_Range,1)

    Pump_Efficiency(index1,:,:) = Pump_Efficiency_PSD_2D( ...
            index1,...
            Pump.Pressure_Range, ...
            Pump.Speed_Range, ...
```

```
            Pump.Torque_Range, ...
            Pump.Displacement_Range, ...
            Pump.Displacement_Map(index1,:,:), ...
            Pump.Efficiency_Map(index1,:,:));

end
Pump.Efficiency = Pump_Efficiency;
clear Pump_Efficiency
toc

if (Enable_Parallel_Processing)
    matlabpool Cores
end

%Initialize vars and extract data from structures
Pump_Flow_Rate( ...
        1:size(Pump.Pressure_Range,1), ...
        1:size(Pump.Speed_Range,2), ...
        1:size(Pump.Torque_Range,2)) = [0];

tic
Pump_Displacement_Map = Pump.Displacement_Map;
Pump_Vol_Efficiency_Map = Pump.Vol_Efficiency_Map;
Pump_Pressure_Range =  Pump.Pressure_Range;
Pump_Speed_Range = Pump.Speed_Range;
Pump_Torque_Range = Pump.Torque_Range;
Pump_Displacement_Range = Pump.Displacement_Range;
Pump_Displacement_Max = Pump.Displacement_Max;

%The pump flow rate is calculated with respect to pressure, speed, and
%torque
parfor index1 = 1:size(Pump.Pressure_Range,1)

    Pump_Flow_Rate(index1,:,:) = Pump_Flow_Rate_PST( ...
            Pump_Displacement_Map(index1,:,:), ...
            Pump_Vol_Efficiency_Map(index1,:,:), ...
            Pump_Pressure_Range, ...
            Pump_Speed_Range, ...
            Pump_Torque_Range, ...
            Pump_Displacement_Range, ...
            Pump_Displacement_Max);

end
Pump.Flow_Rate_Map = Pump_Flow_Rate;
clear Pump_Flow_Rate Pump_Displacement_Map Pump_Vol_Efficiency_Map ...
        Pump_Pressure_Range Pump_Speed_Range Pump_Torque_Range ...
        Pump_Displacement_Range Pump_Displacement_Max

toc

if (Enable_Parallel_Processing)
    matlabpool close
end
```

```
%% Retrieve Motor Data
display('Retrieving Motor Data');
[Motor.Efficiency_Map ...
    Motor.Vol_Efficiency_Map ...
    Motor.Mech_Efficiency_Map ...
    Motor.Displacement_Map ...
    Motor.Pressure_Range ...
    Motor.Displacement_Divs ...
    Motor.Torque_Range ...
    Motor.Displacement_Max] ...
        = P14_Motor(Motor.Speed_Range,Motor.Pressure_Increment);

Motor.Displacement_Range = ...
        (0:Motor.Displacement_Divs)/Motor.Displacement_Divs;


%% Motor Calcs
%An efficiency array with axes of pressure, speed, and torque is created
%for the motor
tic
Efficiency( ...
        1:size(Motor.Pressure_Range,1), ...
        1:size(Motor.Speed_Range,2), ...
        1:size(Motor.Torque_Range,2))=0;

display('Creating Motor Efficiency Array')
for index1 = 1:size(Motor.Pressure_Range,1)

    Efficiency(index1,:,:) = Motor_Efficiency_PSD_2D( ...
            index1, ...
            Motor.Pressure_Range, ...
            Motor.Speed_Range, ...
            Motor.Displacement_Range, ...
            Motor.Torque_Range, ...
            Motor.Displacement_Map(index1,:,:), ...
            Motor.Efficiency_Map(index1,:,:));

end
Motor.Efficiency = Efficiency;
clear Efficiency
toc


%% Hydrostatic Transmission
display('Hydrostatic Transmission');
Hydrostatic_Transmission


%% Hybrid Transmission
display('Hybrid Transmission');
Hybrid_Transmission


%% Retrieve and Process Schedules
```

```
display('Retrieving Schedules');
[Schedule.FTP.Cycle ...
    Schedule.SC03.Cycle ...
    Schedule.UDDS.Cycle ...
    Schedule.US06.Cycle ...
    Schedule.HWYCOL.Cycle] = Driving_Cycles();
```

Appendix D: Perkins.m

```
%Drew Lozano
%3/6/08
%
%Digalog Data Processing
%
%©2008

%clear everything
% clear all;
% clc;
%close all
%

%setup**********************************************************************
function [BSFC_Map] = Perkins(Speed_Range, Torque_Range)

%% run
%filename
% file = 'Data1.prn';
% file2 = 'Data3.prn';
file3 = 'TC1_Baseline_Engine_Map_001.prn';
%struct for data
diesel = struct;
Willans = struct;
%sets the RPM and Torque tolerances for data parsing
RPM_Tolerance = 50;
Torque_Tolerance = 22;

%Data Limits
RPM_Minimum = 500;
Torque_Minimum = 50;
Sample_Minimum = 50;
Sample_Maximum = 200;

%constants
Displacement = 6;           %[L]
Stroke = 5.0;               %[in]
Bore = 3.937;               %[in]
CompressionRatio = 16.0;    %[-]
Cylinders = 6;              %[-]
Q_lhv = 18000;              %[btu/lbm]
Ambient_Pressure = 14.7;    %[Psi]
Dew_Point = 14.6999;        %[°C]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Import Data%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%reads in Perkins DAQ files, removes empty line at end, transposes data

% data1 = dlmread(file,'\t',10,3);
% data2 = dlmread(file2,'\t',10,3);
%
% data = [data1 data2];
```

```
data = dlmread(file3,'\t',[10 3 26 14002]);

%data(18,:) = [];
data = transpose(data);
%stores dimensions of the data array
data_dims = size(data);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Data Format%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Data Array Column Headings
% 1 - Torque                              [ft-lbf]
% 2 - Dyno Speed                          [RPM]
% 3 - Engine Coolant Temperature          [F]
% 4 - Engine Oil Pressure                 [psi]
% 5 - Engine Oil Temperature              [F]
% 6 - Fuel Temperature                    [F]
% 7 - Fuel Mass Flow Rate                 [kg/hr]
% 8 - Ambient Air Temperature             [F]
% 9 - Intercooler Charge Temperature      [F]
% 10 - Intake Charge Pressure             [psi]
% 11 - Intake Manifold Temperature        [F]
% 12 - Intercooler Coolant Inlet Temperature [F]
% 13 - Intercooler Coolant Outlet Temperature [F]
% 14 - Exhaust Manifold Temperature       [F]
% 15 - Exhaust Manifold Pressure          [psi]
% 16 - Exhaust Temperature                [F]
% 17 - Laminar Flow Element Meter         [CFM]
% CALCS
% 18 - Power                              [hp]
% 19 - BSFC                               [lbm/(hp hr)]
% 20 - Fuel Conversion Efficiency         [%]
% 21 - Intercooler Effectiveness          [-]
% 22 - Atmospheric Volumetric Efficiency [%]
% 23 - Manifold Volumetric Efficiency     [%]
% 24 - Air Mass Flow Rate                 [lbm/min]
% 25 - Mean Piston Speed                  [ft/sec]
% 26 - BMEP                               [psi]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Torque = 1;
Speed = 2;
ECT =   3;
EOP =   4;
EOT =   5;
FT =    6;
FFR =   7;

Base_Index = 1;
Sample_Count = 1;
Data_Set = 1;
flag = true;
bad_data = false;

%Parse Data%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Data is parsed into sets of data based on RPM and Torque values
```

```
%Tolerances are used to define operating points. Small sets of data are
%discarded
while(flag)

    if ( (Base_Index + Sample_Count) < data_dims(1))
        if ( ...
            (data(Base_Index,Torque) > Torque_Minimum) ...
            && ...
            (data(Base_Index,Speed) > RPM_Minimum) ...
            && ...
            (Sample_Count < Sample_Maximum) ...
           )
            Operating_Point(Data_Set,:) = ...
                    [data(Base_Index,Torque) data(Base_Index,Speed)];

            if ( ...
                ( ...
                    ( ...
                        data(Base_Index + Sample_Count,1) ...
                        < ...
                        Operating_Point(Data_Set,Torque) ...
                        + Torque_Tolerance ...
                    ) ...
                    && ...
                    ( ...
                        data(Base_Index + Sample_Count,Torque) ...
                        > ...
                        Operating_Point(Data_Set,Torque) ...
                        - Torque_Tolerance ...
                    ) ...
                ) ...
                && ...
                ( ...
                    ( ...
                        data(Base_Index + Sample_Count,Speed) ...
                        < ...
                        Operating_Point(Data_Set,Speed) ...
                        + RPM_Tolerance ...
                    ) ...
                    && ...
                    ( ...
                        data(Base_Index + Sample_Count,Speed) ...
                        > ...
                        Operating_Point(Data_Set,Speed) ...
                        - RPM_Tolerance ...
                    ) ...
                ) ...
               )
                Sample_Count = Sample_Count +1;
                if (data(Base_Index + Sample_Count,FFR) < 0)
                    bad_data = true;
                end

            else
                if ((Sample_Count >= Sample_Minimum) && (bad_data == false))
                    diesel(Data_Set).data = ...
```

```
                            data(Base_Index:Base_Index+Sample_Count-1,:);
                    diesel(Data_Set).samples = Sample_Count;
                    Data_Set = Data_Set + 1;
                else
                    bad_data = false;
                end
                Base_Index = Base_Index + Sample_Count;
                Sample_Count = 1;

            end
        elseif (Sample_Count == 1)
            Base_Index = Base_Index+1;
        else
            if ((Sample_Count >= Sample_Minimum) && (bad_data == false))
                diesel(Data_Set).data = ...
                        data(Base_Index:Base_Index+Sample_Count-1,:);
                diesel(Data_Set).samples = Sample_Count;
                Data_Set = Data_Set + 1;
            else
                bad_data = false;
            end
            Base_Index = Base_Index + Sample_Count;
            Sample_Count = 1;

        end
    else
        diesel(Data_Set).data = data(Base_Index:Base_Index+Sample_Count-1,:);
        diesel(Data_Set).samples = Sample_Count;
        Base_Index = Base_Index + Sample_Count;
        flag = false;
    end


end

diesel_dims = size(diesel);

%Calculations%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Power Calculations
for count = 1:diesel_dims(2)
    for count2 = 1:diesel(count).samples
        diesel(count).calcs(count2,1) = diesel(count).data(count2,1) ...
                * diesel(count).data(count2,2) / 5252;
    end
end

%BSFC Calculations
for count = 1:diesel_dims(2)
    for count2 = 1:diesel(count).samples
        diesel(count).calcs(count2,2) = diesel(count).data(count2,7) ...
                * 2.204624418059/ diesel(count).calcs(count2,1);
    end
end
```

```
%Fuel Conversion Efficiency Calculations
for count = 1:diesel_dims(2)
    for count2 = 1:diesel(count).samples
        diesel(count).calcs(count2,3) = 2545 ...
                / (diesel(count).calcs(count2,2) * Q_lhv);
    end
end

%Intercooler Effectiveness Calculations
for count = 1:diesel_dims(2)
    for count2 = 1:diesel(count).samples
        diesel(count).calcs(count2,4) = (diesel(count).data(count2,9) ...
                - diesel(count).data(count2,11)) ...
                / (diesel(count).data(count2,9) ...
                - diesel(count).data(count2,12));
    end
end

%Atmospheric Volumetric Efficiency Calculations
for count = 1:diesel_dims(2)
    for count2 = 1:diesel(count).samples
        diesel(count).calcs(count2,5) = 2 * diesel(count).data(count2,17) ...
                / (0.035314666919 * Displacement ...
                * diesel(count).data(count2,2));
    end
end

%Inlet Volumetric Efficiency Calculations
for count = 1:diesel_dims(2)
    for count2 = 1:diesel(count).samples
        diesel(count).calcs(count2,6) = 2304 ...
                * diesel(count).data(count2,17) ...
                / (Stroke * Bore^2 * pi * diesel(count).data(count2,2)) ...
                * ( ...
                    Ambient_Pressure ...
                    / (Ambient_Pressure + diesel(count).data(count2,10)) ...
                ) ...
                * (diesel(count).data(count2,11) + 459.67) ...
                / (diesel(count).data(count2,8) + 459.67);
    end
end

%Air Mass Flow Rate Calculations
for count = 1:diesel_dims(2)
    for count2 = 1:diesel(count).samples
        diesel(count).calcs(count2,7) = 2.699504268 ...
                * diesel(count).data(count2,17) * Ambient_Pressure ...
                / ( diesel(count).data(count2,8) + 459.67);
    end
end

%Mean Piston Speed Calculations
for count = 1:diesel_dims(2)
    for count2 = 1:diesel(count).samples
        diesel(count).calcs(count2,8) = 2 /60 ...
                * (Stroke/12) * diesel(count).data(count2,2);
```

```
    end
end

%BMEP Calculations
for count = 1:diesel_dims(2)
    for count2 = 1:diesel(count).samples
        diesel(count).calcs(count2,9) = diesel(count).calcs(count2,1) ...
                * 2 * 396000 / (6 * Stroke * pi * (Bore^2 /4) ...
                * diesel(count).data(count2,2));
    end
end


%Calculate average data for sets of data
for count = 1:diesel_dims(2)
    for count2 = 1:size(diesel(1).data,2)
        diesel(count).average(count2) = mean(diesel(count).data(:,count2));
    end
    for count2 = (size(diesel(1).data,2)+1):...
                (size(diesel(1).data,2)+size(diesel(1).calcs,2))

        diesel(count).average(count2) = ...
                mean(diesel(count).calcs(:,count2-17));
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


for index = 1:size(diesel,2)
    averaged_data(:,index) = diesel(index).average(:);
end


BSFC_Map = griddata( ...
        averaged_data(Speed,:), ...
        averaged_data(Torque,:), ...
        averaged_data(19,:), ...
        Speed_Range, ...
        (Torque_Range)', ...
        'linear');

%% Extrapolate BSFC data at low torques
low_torque = find(Torque_Range >= 51, 1);
high_torque = find(Torque_Range >= 61, 1);
low_speed = find(Speed_Range >= 1200, 1);
high_speed = low_speed + 1;
BSFC_Map(1:low_torque,1:low_speed) = NaN;
BSFC_Map(1:high_torque,high_speed:length(Speed_Range)) = NaN;
for index1 = 1:low_torque
    for index2 = 1:low_speed
    count = low_torque+1-index1;
    BSFC_Map(count,index2) = interp1( ...
            Torque_Range, ...
            BSFC_Map(:,index2), ...
            Torque_Range(count), ...
```

```
            'cubic','extrap');

    end
end
for index1 = 1:high_torque
    for index2 = high_speed:length(Speed_Range)
    count = high_torque+1-index1;
    BSFC_Map(count,index2) = interp1( ...
            Torque_Range, ...
            BSFC_Map(:,index2), ...
            Torque_Range(count), ...
            'cubic','extrap');

    end
end
```

Appendix E: P11_Pump.m

```
%% P11_Pump.m
% Data for a P11 Pump

function [Overall_Efficiency, ...
        Volumetric_Efficiency, ...
        Mechanical_Efficiency, ...
        Displacement_Map, ...
        Pressure_Points, ...
        displacement_divs, ...
        Displacement] = P11_Pump(Speed_Range,Torque_Range,Pressure_Increment)


displacement_divs = 50;

Displacement = 180/2.54^3;%in^3

Speed_Points = Speed_Range;
Pressure_Points = [1000:Pressure_Increment:6000]';
Torque_Points = Torque_Range;

%P11 Performance Data
%Catalog LT3-00010-1-A_Gold_Cup.pdf
%Page 58
Speed_Points_Datasheet = [];
Pressure_Points_Datasheet = [];
Base_Pressure = 1000;
Pressure_Step = 1000;
for index=1:6
    Speed_Points_Datasheet = ...
        [Speed_Points_Datasheet 900  1200   1800    2400];
    Pressure_Points_Datasheet = [Pressure_Points_Datasheet ...
            (Base_Pressure+Pressure_Step*(index-1)) ...
            (Base_Pressure+Pressure_Step*(index-1))  ...
            (Base_Pressure+Pressure_Step*(index-1)) ...
            (Base_Pressure+Pressure_Step*(index-1))];
end


Volumetric_Efficiency_Datasheet = [0.95210566   0.955354717 0.951524528 ...
        0.9527  0.93801761  0.950071698 0.935675472 0.938620755 ...
        0.92392956  0.937735849 0.922186164 0.922322642 0.905157233 ...
        0.920116981 0.903396226 0.902960377 0.889308176 0.895471698 ...
        0.883584906 0.88259434  0.871698113 0.867735849 0.858050314 ...
        0.858160377];

%Calculated from the datasheet. Overall efficiency < 1
Mechanical_Efficiency_Datasheet = [0.857605178  0.821705426 0.817901235 ...
        0.766449747 0.970695971 0.92495637  0.923344948 0.895081275 ...
        1.006329114 0.970695971 0.973668096 0.948970457 1.029126214 ...
        1.003787879 1.000944287 0.981481481 1.043307087 1.018838908 ...
        1.019622932 0.999245852 1.057180851 1.032132425 1.032802858 ...
        1.012174743];
```

```
Volumetric_Efficiency(:,:,1) = griddata( ...
        Speed_Points_Datasheet, ...
        Pressure_Points_Datasheet, ...
        Volumetric_Efficiency_Datasheet, ...
        Speed_Points,Pressure_Points, ...
        'cubic', {'QJ', 'Qbb', 'Qc',});

Mechanical_Efficiency = griddata( ...
        Speed_Points_Datasheet, ...
        Pressure_Points_Datasheet, ...
        Mechanical_Efficiency_Datasheet, ...
        Speed_Points,Pressure_Points, ...
        'cubic', {'QJ', 'Qbb', 'Qc',});

Volumetric_Efficiency = Volumetric_Efficiency(:,:,ones(1,displacement_divs));
displacement_fractions = (1:displacement_divs)/displacement_divs;
displacement_array = permute( ...
        displacement_fractions(ones(1,size(Volumetric_Efficiency,1)), ...
        :, ...
        ones(1,size(Volumetric_Efficiency,2))), ...
        [1 3 2]);

Volumetric_Efficiency = 1 ...
        - (1-Volumetric_Efficiency(:,:,1:displacement_divs)) ...
        ./displacement_array;

pressure_grid = Pressure_Points( ...
        :, ...
        ones(1,size(Volumetric_Efficiency,2)), ...
        ones(1,size(Volumetric_Efficiency,3)));

mechanical_efficiency_grid =  ...
        Mechanical_Efficiency(:,:,ones(1,size(Volumetric_Efficiency,3)));

torque_grid = permute( ...
        Torque_Points(ones(1,size(Volumetric_Efficiency,1)), ...
        :, ...
        ones(1,size(Volumetric_Efficiency,2))), ...
        [1 3 2]);

Overall_Efficiency = Volumetric_Efficiency ...
        .*Mechanical_Efficiency(:,:,ones(1,size(Volumetric_Efficiency,3)));


Torque_Map = Displacement*pressure_grid ...
        .*displacement_array./mechanical_efficiency_grid/(2*pi*12);

Displacement_Map = torque_grid ...
        .*Mechanical_Efficiency(:,:,ones(1,size(torque_grid,3))) ...
        ./ Pressure_Points( ...
            :, ...
            ones(1,size(torque_grid,2)), ...
```

```
            ones(1,size(torque_grid,3)) ...
            ) ...
        * (12*2*pi)/Displacement;


%Remove out of range values
for index1 = 1:size(Displacement_Map,1)
    for index2 = 1:size(Displacement_Map,2)
        for index3 = 1:size(Displacement_Map,3)
            if (Displacement_Map(index1,index2,index3) > 1)
                Displacement_Map(index1,index2,index3) = 0;
            elseif (isnan(Displacement_Map(index1,index2,index3)))
                Displacement_Map(index1,index2,index3) = 0;
            end
        end
    end
end


for index1 = 1:size(Volumetric_Efficiency,1)
    for index2 = 1:size(Volumetric_Efficiency,2)
        for index3 = 1:size(Volumetric_Efficiency,3)
            if (isnan(Volumetric_Efficiency(index1,index2,index3)))
                Volumetric_Efficiency(index1,index2,index3) = 0;
            end
        end
    end
end


for index1 = 1:size(Overall_Efficiency,1)
    for index2 = 1:size(Overall_Efficiency,2)
        for index3 = 1:size(Overall_Efficiency,3)
            if(Overall_Efficiency(index1,index2,index3) < 0)
                Overall_Efficiency(index1,index2,index3) = 0;
            end
        end
    end
end

end
```

Appendix F: Pump_Efficiency_PSD_2D.m

```
%% Pump_Efficiency_PSD_2D
% This function takes an array with axes of speed and
% displacement and turns it into an array with axes of
% speed and torque

function [Pump_Efficiency]= Pump_Efficiency_PSD_2D( ...
        index1, ...
        Pressure_Range, ...
        Speed_Range, ...
        Torque_Range, ...
        Displacement_Range, ...
        Pump_Displacement_Map, ...
        Pump_Efficiency_Map )

    Speed_Map = Speed_Range(ones(1,length(Torque_Range)),:);
    Pump_Efficiency_Map = permute(Pump_Efficiency_Map,[3 2 1]);
    Pump_Displacement_Map = permute(Pump_Displacement_Map,[3 2 1]);

    %Temporarily replace NaNs in the array to prevent errors with interp2
    NaN_Array(1:length(Torque_Range),1:length(Speed_Range)) = 1;
    for index2 = 1:length(Torque_Range)
        for index3 = 1:length(Speed_Range)
            if(isnan(Pump_Displacement_Map(index2,index3)))
                Pump_Displacement_Map(index2,index3) = 1;
                NaN_Array(index2,index3) = NaN;
            end
        end
    end


    Pump_Efficiency = interp2( ...
            Speed_Range, ...
            Displacement_Range, ...
            Pump_Efficiency_Map, ...
            Speed_Map, ...
            Pump_Displacement_Map);


    Pump_Efficiency = permute((Pump_Efficiency.*NaN_Array),[3 2 1]);
    clear Nan_Array

end
```

Appendix G: Pump_Flow_Rate_PST.m

```
%% Pump_Flow_Rate_PST
% This function returns the flow rate of the pump across its speed and
% torque range

function [Flow_Rate] = Pump_Flow_Rate_PST( ...
                            Displacement_Map,...
                            Volumetric_Efficiency_Map, ...
                            Pressure_Range, ...
                            Speed_Range, ...
                            Torque_Range, ...
                            Displacement_Range, ...
                            Displacement_Max)

    Displacement_Map = permute(Displacement_Map, [3 2 1]);
    Volumetric_Efficiency_Map = permute(Volumetric_Efficiency_Map,[3 2 1]);


    Flow_Rate(1,1:size(Speed_Range,2),1:size(Torque_Range,2)) = [0];

    for index1 = 1:size(Speed_Range,2)

            Operating_Point_Disp = Displacement_Map(:,index1);
            Operating_Point_Vol_Efficiency = interp1( ...
                    Displacement_Range, ...
                    Volumetric_Efficiency_Map(:,index1), ...
                    Operating_Point_Disp);

            Flow_Rate(1,index1,:) = permute( ...
                    Displacement_Max*Operating_Point_Disp ...
                    .*Speed_Range(index1) ...
                    .*Operating_Point_Vol_Efficiency/60, ...
                    [3 1 2]);

    end

end
```

Appendix H: P14_Motor.m

```
%% P14_Motor.m
% Data for a P14 hydraulic motor.

function [Overall_Efficiency ...
            Volumetric_Efficiency...
            Mechanical_Efficiency ...
            Displacement_Map ...
            Pressure_Points ...
            displacement_divs ...
            Torque_Points ...
            Displacement] = P14_Motor(Speed_Range,Pressure_Increment)

Displacement = 135/2.54^3;%8; %in^3
displacement_divs = 50;
displacement_fractions = (0:displacement_divs)/displacement_divs;

Torque_Steps=100;

Speed_Points = Speed_Range;
Pressure_Points = [1000:Pressure_Increment:5000]';

Torque_Max = Pressure_Points(size(Pressure_Points,1))*Displacement/(2*12*pi);
Torque_Points = 0:Torque_Max/Torque_Steps:Torque_Max;


%P14 Performance Data
%Catalog LT3-00010-1-A_Gold_Cup.pdf
%Page 59
Speed_Points_Datasheet_Mech = [0    250 400 600 800 1000    1200    1400 ...
                               1600 1800    2000    2200    2400];
Pressure_Points_Datasheet_Mech = [1000 2000 3000 4000 5000];
Pressure_Points_Datasheet_Mech = [(Pressure_Points_Datasheet_Mech(1) ...
        *ones(1,size(Speed_Points_Datasheet_Mech,2))) ...
        (Pressure_Points_Datasheet_Mech(2) ...
        *ones(1,size(Speed_Points_Datasheet_Mech,2))) ...
        (Pressure_Points_Datasheet_Mech(3) ...
        *ones(1,size(Speed_Points_Datasheet_Mech,2))) ...
        (Pressure_Points_Datasheet_Mech(4) ...
        *ones(1,size(Speed_Points_Datasheet_Mech,2))) ...
        (Pressure_Points_Datasheet_Mech(5) ...
        *ones(1,size(Speed_Points_Datasheet_Mech,2)))];

Speed_Points_Datasheet_Mech = repmat(Speed_Points_Datasheet_Mech,1,5);

Speed_Points_Datasheet_Vol = [0:5:2400];
Pressure_Points_Datasheet_Vol = [1000 5000];
Pressure_Points_Datasheet_Vol = [(Pressure_Points_Datasheet_Vol(1) ...
        *ones(1,size(Speed_Points_Datasheet_Vol,2))) ...
        (Pressure_Points_Datasheet_Vol(2) ...
        *ones(1,size(Speed_Points_Datasheet_Vol,2)))];

Speed_Points_Datasheet_Vol = repmat(Speed_Points_Datasheet_Vol,1,2);
```

```
%Calculated from the datasheet. Overall efficiency is < 1
Mechanical_Efficiency_Datasheet = [1.0249   1.002532907 0.992846599 ...
        0.976702752 0.963787674 0.947643827 0.915356133 0.879839669 ...
        0.850780744 0.824950589 0.795891664 0.770061509 0.731316275 ...
        1.0079  0.998496946 0.995268176 0.988810637 0.979124329 ...
        0.969438021 0.958137328 0.946836635 0.932307172 0.921006479 ...
        0.906477017 0.891947555 0.875803707 1.0148  1.007376062 ...
        1.004147292 0.99876601  0.992308471 0.982622163 0.971859598 ...
        0.96217329  0.951410725 0.941724417 0.928809339 0.914818005 ...
        0.90082667  1.006   1.01181562  1.01181562  1.01181562  ...
        1.003743696 0.99486458  0.985178272 0.976299156 0.966612848 ...
        0.956119347 0.945625846 0.934325153 0.92302446  1.0203  ...
        1.0012414   0.999949892 0.996721122 0.990263584 0.984451799 ...
        0.976702752 0.968307952 0.957975889 0.947643827 0.936666011 ...
        0.925042441 0.913418871];

Volumetric_Efficiency_Datasheet = ...
        [1./(0.0604143135/(14/231)+(2.3400406422./(14/231*(0:5:2400)))) ...
        1./(0.0623035461/(14/231)+(2.4354652625./(14/231*(0:5:2400))))];


%Makes a finer grid
Volumetric_Efficiency = griddata( ...
        Speed_Points_Datasheet_Vol, ...
        Pressure_Points_Datasheet_Vol, ...
        Volumetric_Efficiency_Datasheet, ...
        Speed_Points,Pressure_Points, ...
        'linear', {'QJ', 'Qbb', 'Qc',});

Mechanical_Efficiency = griddata( ...
        Speed_Points_Datasheet_Mech, ...
        Pressure_Points_Datasheet_Mech, ...
        Mechanical_Efficiency_Datasheet, ...
        Speed_Points,Pressure_Points, ...
        'linear', {'QJ', 'Qbb', 'Qc',});

Volumetric_Efficiency = ...
    Volumetric_Efficiency(:,:,ones(1,length(displacement_fractions)));

displacement_array = permute(...
        displacement_fractions(ones(1,size(Volumetric_Efficiency,1)),...
        :, ...
        ones(1,size(Volumetric_Efficiency,2))), ...
        [1 3 2]);

Volumetric_Efficiency = 1 ...
        ./ ( ...
            1 ...
            + ( ...
                1 ...
                ./ Volumetric_Efficiency( ...
                    :, ...
                    :, ...
                    1:length(displacement_fractions) ...
```

```
                    ) ...
                - 1 ...
            ) ...
            ./ displacement_array ...
        );

Overall_Efficiency = Volumetric_Efficiency ...
        .*Mechanical_Efficiency(:,:,ones(1,size(Volumetric_Efficiency,3)));

torque_grid = permute( ...
        Torque_Points(ones(1,size(Volumetric_Efficiency,1)), ...
        :, ...
        ones(1,size(Volumetric_Efficiency,2))), ...
        [1 3 2]);

Displacement_Map = torque_grid ...
        ./ Mechanical_Efficiency(:,:,ones(1,size(torque_grid,3))) ...
        ./Pressure_Points( ...
            :, ...
            ones(1,size(torque_grid,2)), ...
            ones(1,size(torque_grid,3)) ...
            ) ...
        * (12*2*pi) / Displacement;

%Removing out of range values
for index1 = 1:size(Displacement_Map,1)
    for index2 = 1:size(Displacement_Map,2)
        for index3 = 1:size(Displacement_Map,3)
            if (Displacement_Map(index1,index2,index3) > 1)
                Displacement_Map(index1,index2,index3) = NaN;
            elseif (isnan(Displacement_Map(index1,index2,index3)))
                Displacement_Map(index1,index2,index3) = NaN;
            end
        end
    end
end

for index1=1:size(Mechanical_Efficiency,1)
    for index2 = 1:size(Mechanical_Efficiency,2)
        if (isnan(Mechanical_Efficiency(index1,index2)))
            Mechanical_Efficiency(index1,index2)=0;
        end
    end
end

for index1=1:size(Volumetric_Efficiency,1)
    for index2 = 1:size(Volumetric_Efficiency,2)
        for index3 = 1:size(Volumetric_Efficiency,3)
            if (isnan(Volumetric_Efficiency(index1,index2,index3)))
                Volumetric_Efficiency(index1,index2,index3)=0;
            elseif (Volumetric_Efficiency(index1,index2,index3) < 0)
                Volumetric_Efficiency(index1,index2,index3)=0;
            end
        end
    end
end
```

Appendix I: Motor_Efficiency_PSD_2D.m

```
%% Motor_Efficiency_PSD_2D
% This function returns the motor efficiency across its speed and torque
% range


function Efficiency = Motor_Efficiency_PSD_2D( ...
                                        index1, ...
                                        Pressure_Range, ...
                                        Speed_Range, ...
                                        Displacement_Range, ...
                                        Torque_Range, ...
                                        Displacement_Map, ...
                                        Efficiency_Map)


    Efficiency_Map=permute(Efficiency_Map,[3 2 1]);
    Displacement_Map=permute(Displacement_Map,[3 2 1]);
    Speed_Map = Speed_Range(ones(1,length(Torque_Range)),:);

    %find out of range values
    NaN_Array(1:length(Torque_Range),1:length(Speed_Range)) = 1;
    for index2 = 1:length(Torque_Range)
        for index3 = 1:length(Speed_Range)
            if(isnan(Displacement_Map(index2,index3)))
                Displacement_Map(index2,index3) = 1;
                NaN_Array(index2,index3) = NaN;
            end
        end
    end

    Efficiency = interp2( ...
            Speed_Range, ...
            Displacement_Range, ...
            Efficiency_Map, ...
            Speed_Map, ...
            Displacement_Map, ...
            'linear');

    Efficiency = permute((Efficiency.*NaN_Array),[3 2 1]);

end
```

Appendix J: Hydrostatic_Transmission.m

```
%% Hydrostatic_Transmission.m
% This file optimizes the engine, pump, and motor operation for driving the
% vehicle in hydrostatic mode
%
% File Structure
%   -Calculate pump torque and displacement maps with axes of
%       flow rate, pressure, and speed
%   -Calculate Engine BSFC and pump efficiency with respect to flow rate,
%       pressure,speed, and gear ratio
%   -Find the minimum effective BSFC on the speed axis
%   -Interpolate engine and pump parameters along the flow rate axis to
%       align with the motor flow rates and pressures
%   -Combine the engine-pump array with the motor array. 4-D array with
%       axes of motor torque, motor speed, pressure, and engine:pump gear
%       ratio
%   -Find the minimum BSFCs along the pressure axis
%   -Calculate fuel mileage
%   -Find the maximum torque at each motor speed
%
% File Dependencies
%   -Hydrostatic_BSFC_Eta_PQW
%   -Hydrostatic_Effective_BSFC_PQW


%Calculate pump displacement and torque flow rate maps
Hydrostatic.EP_Gear_Selection = 1:length(Vehicle.EP_Gear_Ratios);
Hydrostatic.EP_Gear_Ratios = ...
        Vehicle.EP_Gear_Ratios(Hydrostatic.EP_Gear_Selection);

Pump.Flow_Range = 0:5:Pump.Displacement_Max*max(Pump.Speed_Range)/60;

Pump.Disp_PQW_Map = Pump.Flow_Range( ...
        ones(1,length(Pump.Pressure_Range)), ...
        :, ...
        ones(1,length(Pump.Speed_Range)) ...
        ) ...
        / Pump.Displacement_Max ...
        ./permute(Pump.Speed_Range( ...
            ones(1,length(Pump.Pressure_Range)), ...
            :, ...
            ones(1,length(Pump.Flow_Range))) ...
            ,[1 3 2] ...
        ) ...
        *60 ...
        + ( ...
            1 ...
            - permute( ...
                Pump.Vol_Efficiency_Map( ...
                    :, ...
                    :, ...
                    length(Pump.Displacement_Range) ...
                    *ones(1,length(Pump.Flow_Range))), ...
                    [1 3 2] ...
                ) ...
```

```
        );

%Replace out of range values with NaN
for index1 = 1:size(Pump.Disp_PQW_Map,1)
    for index2 = 1:size(Pump.Disp_PQW_Map,2)
        for index3 = 1:size(Pump.Disp_PQW_Map,3)
            if (Pump.Disp_PQW_Map(index1,index2,index3) > 1)
                Pump.Disp_PQW_Map(index1,index2,index3)=NaN;
            end
        end
    end
end

Pump.Torque_PQW_Map = Pump.Displacement_Max*Pump.Disp_PQW_Map ...
        .*Pump.Pressure_Range( ...
            :, ...
            ones(1,length(Pump.Flow_Range)), ...
            ones(1,length(Pump.Speed_Range)) ...
            ) ...
        ./ permute( ...
            Pump.Mech_Efficiency_Map(:,:,ones(1,length(Pump.Flow_Range))),...
            [1 3 2] ...
            ) ...
        /(12*2*pi);


%%
if (Enable_Parallel_Processing)
matlabpool Cores
end

%Extract data from structures
tic
length_Pump_Pressure_Range = length(Pump.Pressure_Range);
Pump_Torque_PQW_Map = Pump.Torque_PQW_Map;
length_Pump_Flow_Range = length(Pump.Flow_Range);
Vehicle_EP_Gear_Ratios = Vehicle.EP_Gear_Ratios;
Engine_Speed_Range = Engine.Speed_Range;
Engine_Torque_Range = Engine.Torque_Range;
Engine_BSFC_Map = Engine.BSFC_Map;
Pump_Speed_Range = Pump.Speed_Range;
Pump_Torque_Range = Pump.Torque_Range;
Pump_Efficiency = Pump.Efficiency;
Vehicle_Gearbox_Efficiency = Vehicle.Gearbox_Efficiency;

Engine_BSFC_PQW_Map( ...
        1:length(Pump.Pressure_Range), ...
        1:length(Pump.Flow_Range), ...
        1:length(Pump.Speed_Range), ...
        1:length(Vehicle.EP_Gear_Ratios)) = NaN;

Pump_Efficiency_PQW_Map( ...
        1:length(Pump.Pressure_Range), ...
        1:length(Pump.Flow_Range), ...
        1:length(Pump.Speed_Range)) = NaN;
```

```
%Calculate Engine BSFC and pump efficiency with respect to flow rate,
%pressure, speed and gear ratio
%Engine_BSFC_PQW_Map(Pressure,Flow Rate,Pump Speed,Engine:Pump Gear)
%Pump_Efficiency_PQW_Map(Pressure,Flow Rate, Pump Speed)
disp('Engine.BSFC_PQW_Map, Pump.Efficiency_PQW_Map');
parfor index1 = 1:length(Pump.Speed_Range)

    [Engine_BSFC_PQW_Map(:,:,index1,:), ...
        Pump_Efficiency_PQW_Map(:,:,index1)] ...
            =  Hydrostatic_BSFC_Eta_PQW( ...
                length_Pump_Pressure_Range, ...
                Pump_Torque_PQW_Map(:,:,index1),...
                length_Pump_Flow_Range, ...
                Vehicle_EP_Gear_Ratios, ...
                Engine_Speed_Range, ...
                Engine_Torque_Range, ...
                Engine_BSFC_Map, ...
                Pump_Speed_Range, ...
                Pump_Torque_Range, ...
                Pump_Efficiency(:,index1,:), ...
                index1, ...
                Vehicle_Gearbox_Efficiency);

end

Engine.BSFC_PQW_Map = Engine_BSFC_PQW_Map;
Pump.Efficiency_PQW_Map = Pump_Efficiency_PQW_Map;

%cleanup
clear Engine_BSFC_PQW_Map Pump_Efficiency_PQW_Map ...
        length_Pump_Pressure_Range Pump_Torque_PQW_Map ...
        length_Pump_Flow_Range Vehicle_EP_Gear_Ratios Engine_Speed_Range ...
        Engine_Torque_Range Engine_BSFC_Map Pump_Speed_Range ...
        Pump_Torque_Range Pump_Efficiency Vehicle_Gearbox_Efficiency

%Zero efficiency at zero shaft speed
Pump.Efficiency_PQW_Map(:,1,:) = 0;

if (Enable_Parallel_Processing)
matlabpool close
end


%% Effective BSFC at the pump
disp('System.Effective_BSFC_PQW_Map')

System.Effective_BSFC_PQW_Map = Engine.BSFC_PQW_Map ...
        ./Pump.Efficiency_PQW_Map( ...
            :,:,:,ones(1,length(Vehicle.EP_Gear_Ratios)) ...
            );

if (Enable_Parallel_Processing)
```

```
matlabpool Cores
end

System.Effective_BSFC_PQW_Map(:,1,:,:) = NaN;

length_Pump_Pressure_Range = length(Pump.Pressure_Range);
length_Pump_Speed_Range = length(Pump.Speed_Range);
System_Effective_BSFC_PQW_Map = System.Effective_BSFC_PQW_Map;
Pump_Flow_Range = Pump.Flow_Range;

Effective_BSFC_PQW_Map( ...
        1:length(Pump.Pressure_Range), ...
        1, ...
        1:length(Pump.Speed_Range), ...
        1:length(Vehicle.EP_Gear_Ratios))=0;

%Interpolate effective BSFC for 0 flow rate
parfor index3 = 1:length(Vehicle.EP_Gear_Ratios)

    Effective_BSFC_PQW_Map(:,1,:,index3) = ...
            Hydrostatic_Effective_BSFC_PQW( ...
                length_Pump_Pressure_Range, ...
                length_Pump_Speed_Range, ...
                System_Effective_BSFC_PQW_Map(:,:,:,index3), ...
                Pump_Flow_Range);

end
System.Effective_BSFC_PQW_Map(:,1,:,:)= Effective_BSFC_PQW_Map;
%cleanup
clear Effective_BSFC_PQW_Map length_Pump_Pressure_Range ...
        length_Pump_Speed_Range System_Effective_BSFC_PQW_Map Pump_Flow_Range


if (Enable_Parallel_Processing)
matlabpool close
end

%% Find minimum BSFC on the speed axis
display('Calculating Peak EP Efficiency Operating Points')

% The mimimum BSFC is found at each pressure and flow rate for each gear
% ratio
for index1 = 1:length(Pump.Pressure_Range)
    for index2 = 1:length(Pump.Flow_Range)
        [System.BSFC_Hydrostatic_Min(index1,index2,:) Speed_Index] = ...
                min(System.Effective_BSFC_PQW_Map(index1,index2,:,:));

        System.Hydrostatic_Engine_Speed(index1,index2,:) = ...
                Pump.Speed_Range(Speed_Index).*Vehicle.EP_Gear_Ratios;

        System.Hydrostatic_Engine_Torque(index1,index2,:) = ...
                Pump.Torque_PQW_Map(index1,index2,Speed_Index) ...
                ./permute(Vehicle.EP_Gear_Ratios,[1 3 2]) ...
                ./Vehicle.Gearbox_Efficiency;
```

```
        System.Hydrostatic_Pump_Disp(index1,index2,:) = ...
                Pump.Disp_PQW_Map(index1,index2,Speed_Index);

        System.Hydrostatic_Pump_Efficiency(index1,index2,:) = ...
                Pump.Efficiency_PQW_Map(index1,index2,Speed_Index);

        System.Hydrostatic_Pump_Mech_Efficiency(index1,index2,:) =  ...
                Pump.Mech_Efficiency_Map(index1,Speed_Index);

        for index3 = 1:length(Vehicle.EP_Gear_Ratios)
        System.Hydrostatic_Engine_BSFC(index1,index2,index3) = ...
                Engine.BSFC_PQW_Map( ...
                    index1,index2,Speed_Index(index3),index3);

        System.Hydrostatic_Pump_Vol_Efficiency(index1,index2,index3) = ...
                interp1( ...
                    Pump.Displacement_Range, ...
                    permute( ...
                        Pump.Vol_Efficiency_Map( ...
                            index1,Speed_Index(index3),:), ...
                        [3 2 1] ...
                        ), ...
                    System.Hydrostatic_Pump_Disp(index1,index2,index3));
        end
    end
end

clear Speed_Index

%% Interpolate engine and pump parameters to align with the motor array
%Initialize variables
Hydrostatic_Motor_Speed( ...
        1:length(Motor.Torque_Range), ...
        1:length(Motor.Speed_Range), ...
        1:length(Motor.Pressure_Range)) ...
            = Motor.Speed_Range( ...
                ones(1,length(Motor.Torque_Range)), ...
                :, ...
                ones(1,length(Motor.Pressure_Range)));

Hydrostatic_Motor_Torque( ...
        1:length(Motor.Torque_Range), ...
        1:length(Motor.Speed_Range), ...
        1:length(Motor.Pressure_Range)) ...
            = permute( ...
                Motor.Torque_Range(ones(1,length(Motor.Speed_Range)), ...
                :, ...
                ones(1,length(Motor.Pressure_Range))), ...
                [2 1 3] ...
                );

Hydrostatic_Pressure( ...
        1:length(Motor.Torque_Range), ...
        1:length(Motor.Speed_Range), ...
```

```
        1:length(Motor.Pressure_Range)) ...
            = permute( ...
                Motor.Pressure_Range(:,ones(1,length(Motor.Torque_Range)),...
                ones(1,length(Motor.Speed_Range))), ...
                [2 3 1]);

Hydrostatic_Motor_Displacement = permute(Motor.Displacement_Map,[3 2 1]);

Hydrostatic_Motor_Flow_Rate(...
        1:length(Motor.Torque_Range), ...
        1:length(Motor.Speed_Range), ...
        1:length(Motor.Pressure_Range)) = 0;

Hydrostatic_Motor_Efficiency( ...
        1:length(Motor.Torque_Range), ...
        1:length(Motor.Speed_Range), ...
        1:length(Motor.Pressure_Range)) = 0;

Hydrostatic_EP_BSFC( ...
        1:length(Motor.Torque_Range), ...
        1:length(Motor.Speed_Range), ...
        1:length(Motor.Pressure_Range), ...
        Hydrostatic.EP_Gear_Selection) = 0;

Hydrostatic_Engine_Speed( ...
        1:length(Motor.Torque_Range), ...
        1:length(Motor.Speed_Range), ...
        1:length(Motor.Pressure_Range), ...
        Hydrostatic.EP_Gear_Selection) = 0;

Hydrostatic_Engine_Torque( ...
        1:length(Motor.Torque_Range), ...
        1:length(Motor.Speed_Range), ...
        1:length(Motor.Pressure_Range), ...
        Hydrostatic.EP_Gear_Selection) = 0;

Hydrostatic_Engine_BSFC( ...
        1:length(Motor.Torque_Range), ...
        1:length(Motor.Speed_Range), ...
        1:length(Motor.Pressure_Range), ...
        Hydrostatic.EP_Gear_Selection) = 0;

Hydrostatic_Pump_Disp( ...
        1:length(Motor.Torque_Range), ...
        1:length(Motor.Speed_Range), ...
        1:length(Motor.Pressure_Range), ...
        Hydrostatic.EP_Gear_Selection) = 0;

Hydrostatic_Pump_Efficiency( ...
        1:length(Motor.Torque_Range), ...
        1:length(Motor.Speed_Range), ...
        1:length(Motor.Pressure_Range), ...
        Hydrostatic.EP_Gear_Selection) = 0;

Hydrostatic_Pump_Mech_Efficiency( ...
```

```
        1:length(Motor.Torque_Range), ...
        1:length(Motor.Speed_Range), ...
        1:length(Motor.Pressure_Range), ...
        Hydrostatic.EP_Gear_Selection) = 0;

Hydrostatic_Pump_Vol_Efficiency( ...
        1:length(Motor.Torque_Range), ...
        1:length(Motor.Speed_Range), ...
        1:length(Motor.Pressure_Range), ...
        Hydrostatic.EP_Gear_Selection) = 0;



for index1 = 1:length(Motor.Pressure_Range)

    for index2 = 1:length(Motor.Speed_Range)
        Hydrostatic_Motor_Flow_Rate(:,index2,index1) = ...
                Motor.Displacement_Max*Motor.Speed_Range(index2) ...
                *Hydrostatic_Motor_Displacement(:,index2,index1) ...
                ./ interp1( ...
                    Motor.Displacement_Range, ...
                    permute( ...
                        Motor.Vol_Efficiency_Map(index1,index2,:),[3 2 1] ...
                        ), ...
                    Hydrostatic_Motor_Displacement(:,index2,index1) ...
                    ) ...
                /60;

        if ( Motor.Torque_Range(1) == 0)
            Hydrostatic_Motor_Flow_Rate(1,index2,index1) = 0;
        end
        if (Motor.Speed_Range(index2) == 0)
            Hydrostatic_Motor_Flow_Rate(:,index2,index1) = 0;
        end

        Hydrostatic_Motor_Efficiency(:,index2,index1) = ...
                interp1( ...
                    Motor.Displacement_Range, ...
                    permute( ...
                        Motor.Efficiency_Map(index1,index2,:),[3 2 1] ...
                        ), ...
                    Hydrostatic_Motor_Displacement(:,index2,index1));


        %Permuting the result of the interpolation of concatenated
        %permuted arrays
        Hydrostatic_Interp = permute( ...
            interp1( ...
                Pump.Flow_Range, ...
                [permute( ...
                    (System.BSFC_Hydrostatic_Min( ...
                        index1,:,Hydrostatic.EP_Gear_Selection)), ...
                    [2 1 3]), ...
                permute( ...
                    (System.Hydrostatic_Engine_Speed( ...
```

```
                        index1,:,Hydrostatic.EP_Gear_Selection)), ...
                    [2 1 3]), ...
                permute( ...
                    (System.Hydrostatic_Engine_Torque( ...
                        index1,:,Hydrostatic.EP_Gear_Selection)), ...
                    [2 1 3]), ...
                permute( ...
                    (System.Hydrostatic_Engine_BSFC( ...
                        index1,:,Hydrostatic.EP_Gear_Selection)), ...
                    [2 1 3]), ...
                permute( ...
                    (System.Hydrostatic_Pump_Disp( ...
                        index1,:,Hydrostatic.EP_Gear_Selection)), ...
                    [2 1 3]), ...
                permute( ...
                    (System.Hydrostatic_Pump_Efficiency( ...
                        index1,:,Hydrostatic.EP_Gear_Selection)), ...
                    [2 1 3]), ...
                permute( ...
                    (System.Hydrostatic_Pump_Mech_Efficiency( ...
                        index1,:,Hydrostatic.EP_Gear_Selection)), ...
                    [2 1 3]), ...
                permute( ...
                    (System.Hydrostatic_Pump_Vol_Efficiency( ...
                        index1,:,Hydrostatic.EP_Gear_Selection)), ...
                    [2 1 3]) ...
                ], ...
                Hydrostatic_Motor_Flow_Rate(:,index2,index1)), ...
            [1 2 4 3]);


        %Extracting data from the previous interpolation
        Hydrostatic_EP_BSFC(:,index2,index1,:) = ...
                Hydrostatic_Interp(:,1,1,:);
        Hydrostatic_Engine_Speed(:,index2,index1,:) = ...
                Hydrostatic_Interp(:,2,1,:);
        Hydrostatic_Engine_Torque(:,index2,index1,:) = ...
                Hydrostatic_Interp(:,3,1,:);
        Hydrostatic_Engine_BSFC(:,index2,index1,:) = ...
                Hydrostatic_Interp(:,4,1,:);
        Hydrostatic_Pump_Disp(:,index2,index1,:) = ...
                Hydrostatic_Interp(:,5,1,:);
        Hydrostatic_Pump_Efficiency(:,index2,index1,:) = ...
                Hydrostatic_Interp(:,6,1,:);
        Hydrostatic_Pump_Mech_Efficiency(:,index2,index1,:) = ...
                Hydrostatic_Interp(:,7,1,:);
        Hydrostatic_Pump_Vol_Efficiency(:,index2,index1,:) = ...
                Hydrostatic_Interp(:,8,1,:);

    end

end


%% Findind the minimum effective BSFC on the pressure axis
% Hydrostatic.array(Motor Torque, Motor Speed, Engine:Pump Gear)
```

```
tic
Hydrostatic_Effective_BSFC = Hydrostatic_EP_BSFC ...
        ./ Hydrostatic_Motor_Efficiency( ...
            :,:,:,ones(1,length(Hydrostatic.EP_Gear_Ratios)));

for index3 = 1:length(Hydrostatic.EP_Gear_Ratios)
    for index1 = 1:length(Motor.Torque_Range)
        for index2 = 1:length(Motor.Speed_Range)
            [Hydrostatic.Effective_BSFC(index1,index2,index3) ...
                Pressure_Index]  ...
                = min(Hydrostatic_Effective_BSFC(index1,index2,:,index3));

            Hydrostatic.Engine_Speed(index1,index2,index3) = ...
                    Hydrostatic_Engine_Speed( ...
                        index1,index2,Pressure_Index,index3);

            Hydrostatic.Motor_Efficiency(index1,index2,index3) = ...
                    Hydrostatic_Motor_Efficiency( ...
                        index1,index2,Pressure_Index);

            Hydrostatic.Engine_Torque(index1,index2,index3) = ...
                    Hydrostatic_Engine_Torque( ...
                        index1,index2,Pressure_Index,index3);

            Hydrostatic.Engine_BSFC(index1,index2,index3) = ...
                    Hydrostatic_Engine_BSFC( ...
                        index1,index2,Pressure_Index,index3);

            Hydrostatic.Pump_Disp(index1,index2,index3) = ...
                    Hydrostatic_Pump_Disp( ...
                        index1,index2,Pressure_Index,index3);

            Hydrostatic.Motor_Flow_Rate(index1,index2,index3) = ...
                    Hydrostatic_Motor_Flow_Rate( ...
                        index1,index2,Pressure_Index);

            Hydrostatic.Pump_Efficiency(index1,index2,index3) = ...
                    Hydrostatic_Pump_Efficiency( ...
                        index1,index2,Pressure_Index,index3);

            Hydrostatic.Pump_Mech_Efficiency(index1,index2,index3) = ...
                    Hydrostatic_Pump_Mech_Efficiency( ...
                        index1,index2,Pressure_Index,index3);

            Hydrostatic.Pump_Vol_Efficiency(index1,index2,index3) = ...
                    Hydrostatic_Pump_Vol_Efficiency( ...
                        index1,index2,Pressure_Index,index3);

            Hydrostatic.Motor_Displacement(index1,index2,index3) = ...
                    Hydrostatic_Motor_Displacement( ...
                        index1,index2,Pressure_Index);

            Hydrostatic.Pressure(index1,index2,index3) = ...
                    Motor.Pressure_Range(Pressure_Index);
```

```
            Hydrostatic.Motor_Mech_Efficiency(index1,index2,index3) = ...
                    Motor.Mech_Efficiency_Map(Pressure_Index,index2);

            Hydrostatic.Motor_Vol_Efficiency(index1,index2,index3) = ...
                    interp1( ...
                        Motor.Displacement_Range, ...
                        permute( ...
                            Motor.Vol_Efficiency_Map( ...
                                Pressure_Index,index2,:), ...
                            [3 2 1]), ...
                        Hydrostatic.Motor_Displacement( ...
                            index1,index2,index3) ...
                        );
        end
    end
end

%% cleanup
clear   Hydrostatic_Motor_Speed   Hydrostatic_Motor_Torque    ...
        Hydrostatic_Pressure    Hydrostatic_Motor_Displacement  ...
        Hydrostatic_Motor_Flow_Rate    Hydrostatic_Motor_Efficiency ...
        Hydrostatic_EP_BSFC Hydrostatic_Engine_Speed    ...
        Hydrostatic_Engine_Torque    Hydrostatic_Engine_BSFC ...
        Hydrostatic_Pump_Disp    Hydrostatic_Pump_Efficiency ...
        Hydrostatic_Pump_Mech_Efficiency    ...
        Hydrostatic_Pump_Vol_Efficiency    Hydrostatic_Interp   ...
        Hydrostatic_Effective_BSFC

toc


%Extrapolate Effective BSFC values for 0 speed and 0 torque
for index1 = 1:length(Hydrostatic.EP_Gear_Ratios)
    Hydrostatic.Effective_BSFC(:,1,index1) = NaN;
    Hydrostatic.Effective_BSFC(1,:,index1) = NaN;

    Hydrostatic.Effective_BSFC( ...
            1,2:size(Hydrostatic.Effective_BSFC,2),index1) ...
                = interp1( ...
                    Motor.Torque_Range, ...
                    Hydrostatic.Effective_BSFC( ...
                        :, ...
                        2:size(Hydrostatic.Effective_BSFC,2), ...
                        index1), ...
                    Motor.Torque_Range(1), ...
                    'cubic','extrap');

    Hydrostatic.Effective_BSFC(:,1,index1) = ...
            interp1( ...
                Motor.Speed_Range, ...
                permute(Hydrostatic.Effective_BSFC(:,:,index1),[2 1]), ...
                Motor.Speed_Range(1), ...
                'cubic','extrap');
end
```

```
%% Calculate the fuel mileage
for index1 = 1:length(Hydrostatic.EP_Gear_Ratios)
Hydrostatic.MPG(:,:,index1) = 7.09*Vehicle.Tire_Diameter*3600*550 ...
        ./ ( ...
            2*5280*Vehicle.Differential_Ratio ...
            .* permute( ...
                Motor.Torque_Range(ones(1,length(Motor.Speed_Range)),:), ...
                [2 1]) ...
            .*Hydrostatic.Effective_BSFC(:,:,index1) ...
        );
end



%% Calculate the maximum torque at each motor speed
for index1 = 1:length(Hydrostatic.EP_Gear_Ratios)
    Torques =  ...
            ( ...
                Hydrostatic.Motor_Displacement(:,:,index1) ...
                - Hydrostatic.Motor_Displacement(:,:,index1) ...
            ) + permute( ...
                Motor.Torque_Range( ...
                    ones(1,length(Motor.Speed_Range)),:), ...
                [2 1]);

    Hydrostatic.Max_Torque(:,index1) = max(Torques);

end
clear Torques
```

Appendix K: Hybrid_Transmission.m

```
%% Hybrid_Transmission.m
% This file optimizes the engine and pump operation for charging the high
% pressure accumulator
%
% File Structure:
%   -Extract data from structures
%   -Initialize variables
%   -Calculate engine parameters across the pump's pressure range to
%       optimize efficiency
%   -Calculate pump parameters based on the engine's parameters
%
% File Dependencies
%   -Hybrid_Operating_Points

Hybrid.Interp3_Range = 10;
%Extract data from structures
%There were memory problems when trying to send fields into the functions
Vehicle_EP_Gear_Ratios = Vehicle.EP_Gear_Ratios;
Engine_Speed_Range = Engine.Speed_Range;
Engine_Torque_Range = Engine.Torque_Range;
Pump_Speed_Range = Pump.Speed_Range;
Pump_Torque_Range = Pump.Torque_Range;
Pump_Efficiency = Pump.Efficiency;
Engine_BSFC_Map = Engine.BSFC_Map;
length_Pump_Pressure_Range = length(Pump.Pressure_Range);
Vehicle_Gearbox_Efficiency = Vehicle.Gearbox_Efficiency;

%Initialize variables
Hybrid_Engine_BSFC ...
        (1:length(Pump.Pressure_Range),1:length(Vehicle_EP_Gear_Ratios)) = 0;

Hybrid_Effective_BSFC ...
        (1:length(Pump.Pressure_Range),1:length(Vehicle_EP_Gear_Ratios)) = 0;

Hybrid_Engine_Speed ...
        (1:length(Pump.Pressure_Range),1:length(Vehicle_EP_Gear_Ratios)) = 0;

Hybrid_Engine_Torque ...
        (1:length(Pump.Pressure_Range),1:length(Vehicle_EP_Gear_Ratios)) = 0;

%Calculate engine parameters
for index1 = 1:length(Pump.Pressure_Range)

    [Hybrid_Engine_BSFC(index1,:) ...
        Hybrid_Effective_BSFC(index1,:) ...
        Hybrid_Engine_Speed(index1,:) ...
        Hybrid_Engine_Torque(index1,:)] = ...
        ...
            Hybrid_Operating_Points( ...
                index1, ...
                Vehicle_EP_Gear_Ratios, ...
                Engine_Speed_Range, ...
                Engine_Torque_Range, ...
                Pump_Speed_Range, ...
```

```
                Pump_Torque_Range, ...
                permute(Pump_Efficiency(index1,:,:),[3 2 1]), ...
                Engine_BSFC_Map, ...
                length_Pump_Pressure_Range, ...
                Vehicle_Gearbox_Efficiency);
end

Hybrid.Engine_BSFC = Hybrid_Engine_BSFC;
Hybrid.Engine_Speed = Hybrid_Engine_Speed;
Hybrid.Engine_Torque = Hybrid_Engine_Torque;
Hybrid.Effective_BSFC = Hybrid_Effective_BSFC;

%cleanup old variables
clear Hybrid_BSFC_Min Hybrid_Engine_Speed Hybrid_Engine_Torque ...
        Vehicle_EP_Gear_Ratios Engine_Speed_Range ...
        Engine_Torque_Range Pump_Speed_Range Pump_Torque_Range ...
        Pump_Efficiency Engine_BSFC_Map ...
        length_Pump_Pressure_Range Vehicle_Gearbox_Efficiency ...
        Hybrid_Engine_BSFC Hybrid_Effective_BSFC

%Calculate pump parameters
Hybrid.Pump_Speed = Hybrid.Engine_Speed ...
        ./Vehicle.EP_Gear_Ratios(ones(1,length(Pump.Pressure_Range)),:);

Hybrid.Pump_Torque = Hybrid.Engine_Torque ...
        .*Vehicle.EP_Gear_Ratios(ones(1,length(Pump.Pressure_Range)),:) ...
        *Vehicle.Gearbox_Efficiency;

for index1 = 1:length(Pump.Pressure_Range)

        Hybrid.Pump_Displacement(index1,:) = interp2( ...
                Pump.Speed_Range, ...
                Pump.Torque_Range, ...
                permute(Pump.Displacement_Map(index1,:,:),[3 2 1]), ...
                Hybrid.Pump_Speed(index1,:), ...
                Hybrid.Pump_Torque(index1,:));

        Hybrid.Pump_Vol_Efficiency(index1,:) = interp2( ...
                Pump.Speed_Range, ...
                Pump.Displacement_Range, ...
                permute(Pump.Vol_Efficiency_Map(index1,:,:),[3 2 1]), ...
                Hybrid.Pump_Speed(index1,:), ...
                Hybrid.Pump_Displacement(index1,:));

end

Hybrid.Pump_Mech_Efficiency = interp2( ...
        Pump.Speed_Range, ...
        Pump.Pressure_Range, ...
        Pump.Mech_Efficiency_Map, ...
        Hybrid.Pump_Speed, ...
        Pump.Pressure_Range(:,ones(1,length(Vehicle.EP_Gear_Ratios))));

Hybrid.Pump_Efficiency = Hybrid.Pump_Mech_Efficiency ...
        .*Hybrid.Pump_Vol_Efficiency;
```

Appendix L: Driving_Cycles.m

```
%% Driving cycles.m
% This function loads and returns the driving cycles


function [data1 data2 data3 data4 data5] = Driving_Cycles()
file1 = 'Driving Cycles\ftpcol.txt'; %Federal Test Procedure Driving Schedule
file2 = 'Driving Cycles\sc03col.txt';%SC03 Driving Cycle
file3 = 'Driving Cycles\uddscol.txt';%Urban Dynamometer Driving Schedule
file4 = 'Driving Cycles\us06col.txt';%US06 Driving Schedule
file5 = 'Driving Cycles\hwycol.txt'; %Highway Driving Schedule


data1 = dlmread(file1,'\t',2,0);
data2 = dlmread(file2,'\t',2,0);
data3 = dlmread(file3,'\t',2,0);
data4 = dlmread(file4,'\t',2,0);
data5 = dlmread(file5,'\t',2,0);


data1(:,3) = 0;
for index1 = 2:size(data1,1)
    data1(index1 - 1,3) = (data1(index1,2) - data1(index1-1,2)) ...
            /(data1(index1,1)-data1(index1-1,1));
end
data1(:,3) = data1(:,3)*5280/3600;
data1(size(data1,1)+1,:) = [data1(size(data1,1),1)+1 0 0];


data2(:,3) = 0;
for index1 = 2:size(data2,1)
    data2(index1 - 1,3) = (data2(index1,2) - data2(index1-1,2)) ...
            /(data2(index1,1)-data2(index1-1,1));
end
data2(:,3) = data2(:,3)*5280/3600;
data2(size(data2,1)+1,:) = [data2(size(data2,1),1)+1 0 0];

data3(:,3) = 0;
for index1 = 2:size(data3,1)
    data3(index1 - 1,3) = (data3(index1,2) - data3(index1-1,2)) ...
            /(data3(index1,1)-data3(index1-1,1));
end
data3(:,3) = data3(:,3)*5280/3600;
data3(size(data3,1)+1,:) = [data3(size(data3,1),1)+1 0 0];

data4(:,3) = 0;
for index1 = 2:size(data4,1)
    data4(index1 - 1,3) = (data4(index1,2) - data4(index1-1,2)) ...
            /(data4(index1,1)-data4(index1-1,1));
end
data4(:,3) = data4(:,3)*5280/3600;
data4(size(data4,1)+1,:) = [data4(size(data4,1),1)+1 0 0];

data4(:,3) = 0;
```

```
for index1 = 2:size(data5,1)
    data5(index1 - 1,3) = (data5(index1,2) - data5(index1-1,2)) ...
            /(data5(index1,1)-data5(index1-1,1));
end
data5(:,3) = data5(:,3)*5280/3600;
data5(size(data5,1)+1,:) = [data5(size(data5,1),1)+1 0 0];
```

Appendix M: BWR_Model.m

```
%% BWR_Model.m
% This function loads the BWR constants and generates the Cv table for
% Nitrogen

BWR.a       =   0.115712;
BWR.A_0     =   136.0430;
BWR.b       =   2.96688E-6;
BWR.B_0     =   0.1454573e-2;
BWR.c       =   3357.732;
BWR.C_0     =   1.0405534E6;
BWR.alpha   =   5.7882567E-9;
BWR.gamma   =   6.755378E-6;
BWR.R       =   296.7551;

display('Calculating Nitrogen Tables')
Nitrogen_Table
```

Appendix N: Nitrogen_Table.m

```
%% Nitrogen_Table.m
% This file generates a table for Nitrogen's Cv
%
% File Structure
%   -Nitrogen constants and data
%   -Curve fit density at each pressure
%   -Calculate ideal gas Cp
%   -Calculate alpha_bar
%   -Calculate ideal has enthalpy
%   -Calculate ideal gas entropy
%   -Calculate alpha_0
%   -Change axes from temperature and pressure to temperature and density
%   -Curve fit alpha_0 and alpha_bar at each density
%   -Use curve fits to smooth the data
%   -Calculate first derivative of alpha_bar and alpha_0 with respect to
%       tau
%   -Curve fit the first derivatives
%   -Smooth data with the curve fits
%   -Calculate second derivative of alpha_bar and alpha_0 with respect to
%       tau
%   -Calculate Cv
%   -Calculate S
%   -Change axes back to temperature and pressure
%% Freezing to 2000K

%constants
Nitrogen.T_c =              126.193;    %K
Nitrogen.T_0 =              298.15;     %K
Nitrogen.Rho_c =            11.177;     %mol/dm^3
Nitrogen.P_reference =      0.101325;   %MPa
Nitrogen.H_0_reference =    8669;       %J/mol
Nitrogen.S_0_reference =    191.502;    %J/mol K
Nitrogen.R =                8.31434;    %J/mol K
Nitrogen.Rho_reference =   Nitrogen.P_reference/Nitrogen.R*1000/Nitrogen.T_0;
%Table 15
%Coefficients for the ideal gas heat capacity
Nitrogen.Cp.N = [-837.079888737
                37.9147114487
                -0.601737844275
                3.50418363823
                -8.74955653028E-06
                1.48958507239E-08
                -2.56370354277E-12
                1.00773735767
                3353.4061];
```

```
%Table 16
%Parameters for the equation of state

Nitrogen.EoS.N = ...
        [0.9499541827   -2.049741504    0.2650110798    -0.3785445194   0.2481718513    -0.1748429008   ...
        0.07311459372   0.1895290433    -0.2046287122   0.638701748     -0.5272986168   0.05551383553   ...
        -0.0281308071   0.007001895093  -0.00081911106396   0.001659823569  -0.04927710927   ...
        0.1138121942    0.05032519699   0.06012817812   -0.09551409802  -0.01100721771  ...
        -0.0001484600538    -0.005806483467 0.06512013679   0.02118354140   0.01284432210 ...
        -0.01054474910];

Nitrogen.EoS.i = [1 1   1   1   2   2   2   2   3   3   3   4   4   4   ...
                  6   6   1   1   2   2   2   2   2   3   4   4   4   4];
Nitrogen.EoS.j = [0.25  1.00    1.50    3.00    0.25    0.50    2.00 ...
                  3.00    0.25    0.50    0.75    1.00    2.00    3.00 ...
                  1.00    2.00    3.00    4.00    1.00    2.00    5.00 ...
                  8.00    20.00   22.00   4.00    6.00    14.00   18.00];
Nitrogen.EoS.l = [0 0   0   0   0   0   0   0   0   0   0   0   0   0 ...
                   0 0   3   3   2   2   2   4   4   3   2   2   4   4];
Nitrogen.EoS.gamma = [0 0   0   0   0   0   0   0   0   0   0   0   0 ...
                      0 0   0   1   1   1   1   1   1   1   1   1   1 ...
                      1 1];
```



```
%% Armed Services Technical Information Agency - 283441
%atm
Nitrogen.Pressure = [ 1.00E+00  2.00E+00    3.00E+00    5.00E+00    7.00E+00    1.00E+01    1.50E+01    ...
                      2.00E+01    2.50E+01    3.00E+01    3.50E+01    4.00E+01    4.50E+01    5.00E+01    ...
                      6.00E+01    7.00E+01    8.00E+01    9.00E+01    1.00E+02    1.20E+02    1.40E+02    ...
                      1.60E+02    1.80E+02    2.00E+02    2.50E+02    3.00E+02    3.50E+02    4.00E+02    ...
                      4.50E+02    5.00E+02    6.00E+02    7.00E+02    8.00E+02    9.00E+02    1.00E+03];

%K
```

```
Nitrogen.Temperature = [250 260 270 280 290 300 320 340 360 380 400 450 500];

%p/p_0
Nitrogen.Density = ...
        [1.0930E+00 1.0507E+00  1.0117E+00  9.7542E-01  9.4168E-01  9.1023E-01  8.5321E-01  8.0294E-01  ...
        7.5826E-01  7.1829E-01  6.8235E-01  6.0648E-01  5.4581E-01;
        2.1876E+00  2.1028E+00  2.0244E+00  1.9515E+00  1.8839E+00  1.8207E+00  1.7064E+00  1.6058E+00  ...
        1.5162E+00  1.4365E+00  1.3643E+00  1.2125E+00  1.0912E+00;
        3.2841E+00  3.1562E+00  3.0380E+00  2.9284E+00  2.8265E+00  2.7316E+00  2.5598E+00  2.4084E+00  ...
        2.2740E+00  2.1538E+00  2.0458E+00  1.8180E+00  1.6360E+00;
        5.4823E+00  5.2670E+00  5.0682E+00  4.8841E+00  4.7133E+00  4.5541E+00  4.2664E+00  4.0132E+00  ...
        3.7886E+00  3.5880E+00  3.4077E+00  3.0279E+00  2.7245E+00;
        7.6872E+00  7.3827E+00  7.1021E+00  6.8426E+00  6.6019E+00  6.3779E+00  5.9732E+00  5.6174E+00  ...
        5.3022E+00  5.0208E+00  4.7680E+00  4.2359E+00  3.8112E+00;
        1.1006E+01  1.0566E+01  1.0160E+01  9.7849E+00  9.3474E+00  9.1149E+00  8.5330E+00  8.0225E+00  ...
        7.5703E+00  7.1671E+00  6.8048E+00  6.0442E+00  5.4377E+00;
        1.6571E+01  1.5894E+01  1.5272E+01  1.4699E+01  1.4171E+01  1.3681E+01  1.2798E+01  1.2026E+01  ...
        1.1344E+01  1.0737E+01  1.0192E+01  9.0494E+00  8.1396E+00;
        2.2173E+01  2.1250E+01  2.0404E+01  1.9628E+01  1.8912E+01  1.8250E+01  1.7063E+01  1.6025E+01  ...
        1.5110E+01  1.4296E+01  1.3568E+01  1.2042E+01  1.0830E+01;
        2.7806E+01  2.6626E+01  2.5551E+01  2.4565E+01  2.3660E+01  2.2821E+01  2.1322E+01  2.0016E+01  ...
        1.8866E+01  1.7846E+01  1.6933E+01  1.5023E+01  1.3508E+01;
        3.3473E+01  3.2028E+01  3.0715E+01  2.9513E+01  2.8409E+01  2.7391E+01  2.5578E+01  2.3999E+01  ...
        2.2612E+01  2.1384E+01  2.0286E+01  1.7993E+01  1.6176E+01;
        3.9167E+01  3.7445E+01  3.5886E+01  3.4462E+01  3.3161E+01  3.1964E+01  2.9828E+01  2.7973E+01  ...
        2.6348E+01  2.4909E+01  2.3625E+01  2.0948E+01  1.8830E+01;
        4.4897E+01  4.2886E+01  4.1070E+01  3.9422E+01  3.7914E+01  3.6529E+01  3.4069E+01  3.1937E+01  ...
        3.0072E+01  2.8424E+01  2.6953E+01  2.3892E+01  2.1474E+01;
        5.0629E+01  4.8325E+01  4.6250E+01  4.4372E+01  4.2657E+01  4.1092E+01  3.8303E+01  3.5892E+01  ...
        3.3786E+01  3.1927E+01  3.0267E+01  2.6824E+01  2.4105E+01;
        5.6404E+01  5.3803E+01  5.1467E+01  4.9347E+01  4.7415E+01  4.5647E+01  4.2528E+01  3.9836E+01  ...
        3.7483E+01  3.5409E+01  3.3568E+01  2.9740E+01  2.6722E+01;
        6.7931E+01  6.4713E+01  6.1837E+01  5.9237E+01  5.6876E+01  5.4723E+01  5.0929E+01  4.7667E+01  ...
        4.4834E+01  4.2343E+01  4.0128E+01  3.5539E+01  3.1927E+01;
        7.9501E+01  7.5636E+01  7.2199E+01  6.9104E+01  6.6302E+01  6.3755E+01  5.9284E+01  5.5454E+01  ...
        5.2138E+01  4.9217E+01  4.6635E+01  4.1274E+01  3.7073E+01;
        9.1071E+01  8.6533E+01  8.2517E+01  7.8941E+01  7.5713E+01  7.2762E+01  6.7582E+01  6.3180E+01  ...
        5.9363E+01  5.6023E+01  5.3076E+01  4.6958E+01  4.2175E+01;
        1.0258E+02  9.7364E+01  9.2768E+01  8.8692E+01  8.4991E+01  8.1645E+01  7.5790E+01  7.0808E+01  ...
        6.6519E+01  6.2772E+01  5.9441E+01  5.2578E+01  4.7225E+01;
        1.1401E+02  1.0813E+02  1.0296E+02  9.8390E+01  9.4251E+01  9.0483E+01  8.3908E+01  7.8361E+01  ...
```

```
        7.3599E+01  6.9447E+01  6.5757E+01  5.8146E+01  5.2223E+01;
        1.3636E+02  1.2920E+02  1.2296E+02  1.1742E+02  1.1241E+02  1.0786E+02  9.9882E+01  9.3231E+01  ...
        8.7514E+01  8.2548E+01  7.8143E+01  6.9104E+01  6.2060E+01;
        1.5799E+02  1.4966E+02  1.4233E+02  1.3586E+02  1.3003E+02  1.2474E+02  1.1554E+02  1.0776E+02  ...
        1.0110E+02  9.5334E+01  9.0264E+01  7.9813E+01  7.1691E+01;
        1.7866E+02  1.6934E+02  1.6106E+02  1.5366E+02  1.4700E+02  1.4099E+02  1.3056E+02  1.2182E+02  ...
        1.1436E+02  1.0781E+02  1.0207E+02  9.0264E+01  8.1084E+01;
        1.9626E+02  1.8795E+02  1.7880E+02  1.7063E+02  1.6329E+02  1.5667E+02  1.4519E+02  1.3553E+02  ...
        1.2722E+02  1.1993E+02  1.1355E+02  1.0046E+02  9.0264E+01;
        2.1646E+02  2.0535E+02  1.9549E+02  1.8685E+02  1.7894E+02  1.7181E+02  1.5934E+02  1.4876E+02  ...
        1.3969E+02  1.3171E+02  1.2474E+02  1.1036E+02  9.9218E+01;
        2.5757E+02  2.4533E+02  2.3427E+02  2.2430E+02  2.1523E+02  2.0694E+02  1.9234E+02  1.7908E+02  ...
        1.6908E+02  1.5964E+02  1.5129E+02  1.3410E+02  1.2075E+02;
        2.9232E+02  2.7948E+02  2.6773E+02  2.5701E+02  2.4717E+02  2.3811E+02  2.2193E+02  2.0802E+02  ...
        1.9592E+02  1.8529E+02  1.7587E+02  1.5632E+02  1.4109E+02;
        3.2180E+02  3.0880E+02  2.9674E+02  2.8561E+02  2.7533E+02  2.6576E+02  2.4857E+02  2.3359E+02  ...
        2.2049E+02  2.0891E+02  1.9859E+02  1.7713E+02  1.6025E+02;
        3.4707E+02  3.3413E+02  3.2207E+02  3.1077E+02  3.0019E+02  2.9035E+02  2.7252E+02  2.5683E+02  ...
        2.4296E+02  2.3063E+02  2.1960E+02  1.9657E+02  1.7832E+02;
        3.6896E+02  3.5618E+02  3.4419E+02  3.3294E+02  3.2231E+02  3.1229E+02  2.9397E+02  2.7782E+02  ...
        2.6348E+02  2.5063E+02  2.3905E+02  2.1478E+02  1.9532E+02;
        3.8814E+02  3.7552E+02  3.6363E+02  3.5242E+02  3.4183E+02  3.3181E+02  3.1342E+02  2.9701E+02  ...
        2.8230E+02  2.6908E+02  2.5707E+02  2.3180E+02  2.1135E+02;
        4.2088E+02  4.0867E+02  3.9708E+02  3.8613E+02  3.7571E+02  3.6583E+02  3.4745E+02  3.3083E+02  ...
        3.1572E+02  3.0202E+02  2.8949E+02  2.6274E+02  2.4077E+02;
        4.4762E+02  4.3595E+02  4.2479E+02  4.1411E+02  4.0396E+02  3.9429E+02  3.7621E+02  3.5966E+02  ...
        3.4451E+02  3.3058E+02  3.1778E+02  2.9009E+02  2.6706E+02;
        4.6997E+02  4.6625E+02  4.4816E+02  4.3791E+02  4.2812E+02  4.1868E+02  4.0092E+02  3.8461E+02  ...
        3.6957E+02  3.5561E+02  3.4267E+02  3.1448E+02  2.9073E+02;
        4.8937E+02  4.7860E+02  4.6840E+02  4.5852E+02  4.4897E+02  4.3989E+02  4.2255E+02  4.0645E+02  ...
        3.9160E+02  3.7780E+02  3.6488E+02  3.3634E+02  3.1207E+02;
        5.0687E+02  4.9653E+02  4.8650E+02  4.7687E+02  4.6762E+02  4.5871E+02  4.4180E+02  4.2600E+02  ...
        4.1138E+02  3.9772E+02  3.8487E+02  3.5618E+02  3.3161E+02];

Nitrogen.Density_Reference = [1.25046 4.46338E-2]; %[kg/m^3 mol/L]


%% Calculations

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%Density Curve Fitting
%
%The data point at 800 atm, 260K seems off when compared to the rest of the
%data and is excluded from the fit

Nitrogen.Curve_Fit.Density.fittype = fittype('power1');
Nitrogen.Curve_Fit.Density.options = fitoptions('power1');
Nitrogen.Curve_Fit.Density.outlier = ...
    excludedata(Nitrogen.Temperature,Nitrogen.Density(33,:),'indices',[2]);

Nitrogen.Curve_Fit.Density.exclude = fitoptions('power1');
set(Nitrogen.Curve_Fit.Density.options,'MaxFunEvals',2000,'MaxIter',1500,'TolFun',1e-8,'TolX',1e-8);

set( ...
    Nitrogen.Curve_Fit.Density.exclude, ...
    'MaxFunEvals',2000,'MaxIter',1500,'TolFun',1e-8,'TolX',1e-8, ...
    'Exclude',Nitrogen.Curve_Fit.Density.outlier);

for index1 = 1:length(Nitrogen.Pressure)
    if (index1 ~= 33)

        Nitrogen.Curve_Fit.Density.fit = ...
            fit( ...
                Nitrogen.Temperature', ...
                Nitrogen.Density(index1,:)', ...
                Nitrogen.Curve_Fit.Density.fittype, ...
                Nitrogen.Curve_Fit.Density.options);
    else
        Nitrogen.Curve_Fit.Density.fit = ...
                fit( ...
                    Nitrogen.Temperature', ...
                    Nitrogen.Density(index1,:)', ...
                    Nitrogen.Curve_Fit.Density.fittype, ...
                    Nitrogen.Curve_Fit.Density.exclude);
    end

    Nitrogen.Curve_Fit.Density.coeffs(index1,:) = ...
            coeffvalues(Nitrogen.Curve_Fit.Density.fit);
end
```

```
Nitrogen.Density_Filled.Pressure = Nitrogen.Pressure;
Nitrogen.Density_Filled.Temperature_delta = 0.5;
Nitrogen.Density_Filled.Temperature = ...
        Nitrogen.Temperature(1) ...
        :Nitrogen.Density_Filled.Temperature_delta ...
        :Nitrogen.Temperature(length(Nitrogen.Temperature));

for index1 = 1:length(Nitrogen.Pressure)
    Nitrogen.Density_Filled.Val(index1,:) = ...
            Nitrogen.Curve_Fit.Density.coeffs(index1,1) ...
            .*Nitrogen.Density_Filled.Temperature ...
            .^Nitrogen.Curve_Fit.Density.coeffs(index1,2);
end

Nitrogen.Temperature = Nitrogen.Density_Filled.Temperature;
Nitrogen.Density = Nitrogen.Density_Filled.Val;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%Calculate ideal gas Cp
Nitrogen.Cp_0.delta = 0.01;
Nitrogen.Cp_0.Temperature = ...
        Nitrogen.T_c ...
        :Nitrogen.Cp_0.delta ...
        :max(Nitrogen.Temperature)+Nitrogen.Cp_0.delta;

Nitrogen.Cp_0.Val = ...
        Nitrogen.R*(Nitrogen.Cp.N(1)./Nitrogen.Cp_0.Temperature.^3 ...
        + Nitrogen.Cp.N(2)./Nitrogen.Cp_0.Temperature.^2 ...
        + Nitrogen.Cp.N(3)./Nitrogen.Cp_0.Temperature ...
        + Nitrogen.Cp.N(4) ...
        + Nitrogen.Cp.N(5).*Nitrogen.Cp_0.Temperature ...
        + Nitrogen.Cp.N(6).*Nitrogen.Cp_0.Temperature.^2 ...
        + Nitrogen.Cp.N(7).*Nitrogen.Cp_0.Temperature.^3 ...
        + Nitrogen.Cp.N(8).*(Nitrogen.Cp.N(9) ...
        ./Nitrogen.Cp_0.Temperature).^2.*exp(Nitrogen.Cp.N(9) ...
        ./Nitrogen.Cp_0.Temperature) ...
        ./(exp(Nitrogen.Cp.N(9)./Nitrogen.Cp_0.Temperature) - 1).^2);
```

```
%calculate alpha_bar
for index1 = 1:length(Nitrogen.Pressure)
    for index2 = 1:length(Nitrogen.Temperature)
        Nitrogen.alpha_bar(index1,index2) = sum(Nitrogen.EoS.N .* ...
            ( ...
                (Nitrogen.Density_Reference(2)*Nitrogen.Density(index1,index2)/Nitrogen.Rho_c) ...
                .^(Nitrogen.EoS.i) ...
            ) ...
            .* (Nitrogen.T_c/Nitrogen.Temperature(index2)).^(Nitrogen.EoS.j) ...
            .* exp( ...
                -Nitrogen.EoS.gamma ...
                .*(Nitrogen.Density_Reference(2)*Nitrogen.Density(index1,index2)/Nitrogen.Rho_c) ...
                .^(Nitrogen.EoS.l) ...
                ) ...
            );

    end
end


for index1 = 1:length(Nitrogen.Temperature)

    if (Nitrogen.T_0 < Nitrogen.Temperature(index1))
        Nitrogen.int_pos = 1;
        Nitrogen.Bound_1 = find(Nitrogen.Cp_0.Temperature >= Nitrogen.T_0);
        Nitrogen.Bound_2 = find(Nitrogen.Cp_0.Temperature >= Nitrogen.Temperature(index1));
    elseif (Nitrogen.T_0 > Nitrogen.Temperature(index1))
        Nitrogen.int_pos = -1;
        Nitrogen.Bound_1 = find(Nitrogen.Cp_0.Temperature >= Nitrogen.Temperature(index1));
        Nitrogen.Bound_2 = find(Nitrogen.Cp_0.Temperature >= Nitrogen.T_0);
    else
        Nitrogen.int_pos = 0;
        Nitrogen.Bound_1 = find(Nitrogen.Cp_0.Temperature >= Nitrogen.T_0);
        Nitrogen.Bound_2 = find(Nitrogen.Cp_0.Temperature >= Nitrogen.T_0);
    end

    %ideal gas enthalpy
    Nitrogen.H_0(index1) = Nitrogen.H_0_reference ...
            + Nitrogen.int_pos *trapz(Nitrogen.Cp_0.Val(Nitrogen.Bound_1:Nitrogen.Bound_2))...
            *Nitrogen.Cp_0.delta;
```

```
    %ideal has entropy
    Nitrogen.S_0(1:length(Nitrogen.Pressure),index1) = ...
            Nitrogen.S_0_reference ...
            + Nitrogen.int_pos ...
            *sum( ...
                Nitrogen.Cp_0.Val(Nitrogen.Bound_1:Nitrogen.Bound_2) ...
                ./Nitrogen.Cp_0.Temperature(Nitrogen.Bound_1:Nitrogen.Bound_2) ...
                ) ...
            *Nitrogen.Cp_0.delta;

    for index2 = 1:length(Nitrogen.Pressure)
        Nitrogen.S_0(index2,index1) = Nitrogen.S_0(index2,index1) ...
                - Nitrogen.R ...
                * log( ...
                    Nitrogen.Density(index2,index1) ...
                    *Nitrogen.Density_Reference(2)/Nitrogen.Rho_reference ...
                    *Nitrogen.Temperature(index1)/Nitrogen.T_0 ...
                    );
    end

end

Nitrogen.int_tau_delta = 5e-6;

%calculate alpha_0
for index1 = 1:length(Nitrogen.Temperature)

    if (Nitrogen.T_0 < Nitrogen.Temperature(index1))
        Nitrogen.int_pos = -1;
        Nitrogen.int_tau_range = ...
                Nitrogen.T_c/Nitrogen.Temperature(index1):Nitrogen.int_tau_delta:Nitrogen.T_c/Nitrogen.T_0;
    elseif (Nitrogen.T_0 > Nitrogen.Temperature(index1))
        Nitrogen.int_pos = 1;
        Nitrogen.int_tau_range = ...
                Nitrogen.T_c/Nitrogen.T_0:Nitrogen.int_tau_delta:Nitrogen.T_c/Nitrogen.Temperature(index1);
    else
        Nitrogen.int_pos = 0;
```

```
    Nitrogen.int_tau_range = Nitrogen.T_c/Nitrogen.T_0;
end

Nitrogen.alpha_0(1:length(Nitrogen.Pressure),index1) = ...
    Nitrogen.H_0_reference/Nitrogen.R/Nitrogen.Temperature(index1) ...
    - 1 ...
    - Nitrogen.T_c / Nitrogen.Temperature(index1) / Nitrogen.R ...
    * Nitrogen.int_pos ...
    * sum( ...
    Nitrogen.R * ...
    ( ...
        Nitrogen.Cp.N(1)*Nitrogen.int_tau_range.^3/Nitrogen.T_c^3 ...
        + Nitrogen.Cp.N(2)*Nitrogen.int_tau_range.^2/Nitrogen.T_c^2 ...
        + Nitrogen.Cp.N(3)*Nitrogen.int_tau_range/Nitrogen.T_c ...
        + Nitrogen.Cp.N(4) ...
        + Nitrogen.Cp.N(5)*Nitrogen.T_c./Nitrogen.int_tau_range ...
        + Nitrogen.Cp.N(6).*Nitrogen.T_c^2./Nitrogen.int_tau_range.^2 ...
        + Nitrogen.Cp.N(7).*Nitrogen.T_c^3./Nitrogen.int_tau_range.^3 ...
        + Nitrogen.Cp.N(8).*(Nitrogen.Cp.N(9)*Nitrogen.int_tau_range/Nitrogen.T_c).^2 ...
        .*exp(Nitrogen.Cp.N(9)*Nitrogen.int_tau_range/Nitrogen.T_c) ...
        ./(exp(Nitrogen.Cp.N(9)*Nitrogen.int_tau_range/Nitrogen.T_c) - 1).^2 ...
    ) ...
    ./ Nitrogen.int_tau_range.^2) * Nitrogen.int_tau_delta ...
    ...
    + 1 / Nitrogen.R ...
    * Nitrogen.int_pos ...
    * sum( ...
        Nitrogen.R * ...
        ( ...
            Nitrogen.Cp.N(1)*Nitrogen.int_tau_range.^3/Nitrogen.T_c^3 ...
            + Nitrogen.Cp.N(2)*Nitrogen.int_tau_range.^2/Nitrogen.T_c^2 ...
            + Nitrogen.Cp.N(3)*Nitrogen.int_tau_range/Nitrogen.T_c ...
            + Nitrogen.Cp.N(4) ...
            + Nitrogen.Cp.N(5)*Nitrogen.T_c./Nitrogen.int_tau_range ...
            + Nitrogen.Cp.N(6).*Nitrogen.T_c^2./Nitrogen.int_tau_range.^2 ...
            + Nitrogen.Cp.N(7).*Nitrogen.T_c^3./Nitrogen.int_tau_range.^3 ...
            + Nitrogen.Cp.N(8).*(Nitrogen.Cp.N(9)*Nitrogen.int_tau_range/Nitrogen.T_c).^2 ...
            .*exp(Nitrogen.Cp.N(9)*Nitrogen.int_tau_range/Nitrogen.T_c) ...
            ./(exp(Nitrogen.Cp.N(9)*Nitrogen.int_tau_range/Nitrogen.T_c) - 1).^2 ...
        ) ...
        ./ Nitrogen.int_tau_range ....
```

```
          ) ...
        * Nitrogen.int_tau_delta ...
        - Nitrogen.S_0_reference/Nitrogen.R ...
        + log( ...
            Nitrogen.Density(:,index1)*Nitrogen.Density_Reference(2) ...
            ./Nitrogen.Rho_reference*Nitrogen.Temperature(index1)./Nitrogen.T_0 ...
            );

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Define new axes

Nitrogen.Tau_gd_delta = -5e-3;
Nitrogen.Tau_gd = ...
        Nitrogen.T_c/Nitrogen.Temperature(1) ...
        :Nitrogen.Tau_gd_delta ...
        :Nitrogen.T_c/Nitrogen.Temperature(length(Nitrogen.Temperature));

Nitrogen.Temperature_gd = Nitrogen.T_c./Nitrogen.Tau_gd;


Nitrogen.Density_Min = min(Nitrogen.Density);
Nitrogen.Density_Min = min(Nitrogen.Density_Min);
Nitrogen.Density_Max = max(Nitrogen.Density);
Nitrogen.Density_Max = max(Nitrogen.Density_Max);
Nitrogen.Density_Steps = 200;
Nitrogen.Density_Range = ...
        10.^(log10(Nitrogen.Density_Min) ...
        :log10(Nitrogen.Density_Max/Nitrogen.Density_Min)/Nitrogen.Density_Steps ...
        :log10(Nitrogen.Density_Max));
%% Change axes to temperature and density
Nitrogen.alpha_bar_reshape = reshape(permute(Nitrogen.alpha_bar,[2 1]),1,numel(Nitrogen.alpha_bar));
Nitrogen.alpha_0_reshape = reshape(permute(Nitrogen.alpha_0,[2 1]),1,numel(Nitrogen.alpha_bar));
Nitrogen.Temperature_reshape = ...
        reshape( ...
            permute(Nitrogen.Temperature(ones(1,length(Nitrogen.Pressure)),:),[2 1]), ...
            1, ...
            numel(Nitrogen.alpha_bar));
```

```
Nitrogen.Density_reshape = reshape(permute(Nitrogen.Density,[2 1]),1,numel(Nitrogen.Density));

Nitrogen.Pressure_reshape =  ...
        reshape(Nitrogen.Pressure(ones(1,length(Nitrogen.Temperature)),:),1,numel(Nitrogen.alpha_bar));

Nitrogen.alpha_bar_gd = griddata( ...
        Nitrogen.Temperature_reshape, ...
        Nitrogen.Density_reshape, ...
        Nitrogen.alpha_bar_reshape, ...
        Nitrogen.Temperature_gd, ...
        Nitrogen.Density_Range', ...
        'linear');

Nitrogen.alpha_0_gd = griddata( ...
        Nitrogen.Temperature_reshape, ...
        Nitrogen.Density_reshape, ...
        Nitrogen.alpha_0_reshape, ...
        Nitrogen.Temperature_gd, ...
        Nitrogen.Density_Range', ...
        'linear');

Nitrogen.Pressure_gd = griddata( ...
        Nitrogen.Temperature_reshape, ...
        Nitrogen.Density_reshape, ...
        Nitrogen.Pressure_reshape, ...
        Nitrogen.Temperature_gd, ...
        Nitrogen.Density_Range', ...
        'linear');
%%

%clean griddata results
for index1 = 1:length(Nitrogen.Density_Range)

    if( max(Nitrogen.Pressure_gd(index1,:)) >= max(Nitrogen.Pressure) - 1 )

        if ( find(Nitrogen.Pressure_gd(index1,:) >= max(Nitrogen.Pressure)-1,1) ...
             <  ...
             length(Nitrogen.Temperature_gd))
```

```
            Nitrogen.Curve_Fit.outlier = ...
                    (find(Nitrogen.Pressure_gd(index1,:) >= max(Nitrogen.Pressure)-1,1)) ...
                    :length(Nitrogen.Temperature_gd);

            Nitrogen.Pressure_gd(index1,Nitrogen.Curve_Fit.outlier) = NaN;
            Nitrogen.alpha_bar_gd(index1,Nitrogen.Curve_Fit.outlier) = NaN;
            Nitrogen.alpha_0_gd(index1,Nitrogen.Curve_Fit.outlier) = NaN;

        end
    end
end


Nitrogen.Curve_Fit.alpha_0_gd.nans = isnan(Nitrogen.alpha_0_gd);
Nitrogen.Curve_Fit.alpha_bar_gd.nans = isnan(Nitrogen.alpha_bar_gd);
Nitrogen.Curve_Fit.alpha_0_gd.flag = true;
for index1 = 1:length(Nitrogen.Density_Range)




    if (sum(~Nitrogen.Curve_Fit.alpha_0_gd.nans(index1,:)) >= 10)
        Nitrogen.Curve_Fit.alpha_0_gd.exclude =  ...
                excludedata( ...
                    Nitrogen.Temperature_gd, ...
                    Nitrogen.alpha_0_gd(index1,:), ...
                    'indices', ...
                    nonzeros( ...
                        (1:size(Nitrogen.Curve_Fit.alpha_0_gd.nans,2)) ...
                        .*Nitrogen.Curve_Fit.alpha_0_gd.nans(index1,:) ...
                        ) ...
                    );

        Nitrogen.Curve_Fit.alpha_0_gd.fittype = fittype('power2');
        Nitrogen.Curve_Fit.alpha_0_gd.options = fitoptions('power2');

        if (Nitrogen.Curve_Fit.alpha_0_gd.flag)
            set( ...
```

```
                Nitrogen.Curve_Fit.alpha_0_gd.options, ...
                'MaxFunEvals',10000,'MaxIter',9000,'TolFun',1e-10,'TolX',1e-10, ...
                'Exclude',Nitrogen.Curve_Fit.alpha_0_gd.exclude);
            Nitrogen.Curve_Fit.alpha_0_gd.flag = false;
        else

            set( ...
                Nitrogen.Curve_Fit.alpha_0_gd.options, ...
                'MaxFunEvals',10000,'MaxIter',9000,'TolFun',1e-10,'TolX',1e-10, ...
                'Exclude',Nitrogen.Curve_Fit.alpha_0_gd.exclude, ...
                'StartPoint',Nitrogen.Curve_Fit.alpha_0_gd.coeffs, ...
                'Lower', ...
                Nitrogen.Curve_Fit.alpha_0_gd.coeffs-abs(0.2*Nitrogen.Curve_Fit.alpha_0_gd.coeffs), ...
                'Upper', ...
                Nitrogen.Curve_Fit.alpha_0_gd.coeffs+abs(0.2*Nitrogen.Curve_Fit.alpha_0_gd.coeffs));
        end

        [Nitrogen.Curve_Fit.alpha_0_gd.fit Nitrogen.Curve_Fit.alpha_0_gd.gof_struct(index1,:)]  = ...
            fit( ...
                Nitrogen.Temperature_gd', ...
                Nitrogen.alpha_0_gd(index1,:)', ...
                Nitrogen.Curve_Fit.alpha_0_gd.fittype, ...
                Nitrogen.Curve_Fit.alpha_0_gd.options);

        Nitrogen.Curve_Fit.alpha_0_gd.coeffs = coeffvalues(Nitrogen.Curve_Fit.alpha_0_gd.fit);

        Nitrogen.Curve_Fit.alpha_0_gd.params(index1,:) = ...
                [Nitrogen.Curve_Fit.alpha_0_gd.gof_struct(index1).rsquare ...
                Nitrogen.Curve_Fit.alpha_0_gd.gof_struct(index1).sse ...
                Nitrogen.Curve_Fit.alpha_0_gd.coeffs];

        %smooth the data
        Nitrogen.alpha_0_gd_fit(index1,:) =  Nitrogen.Curve_Fit.alpha_0_gd.coeffs(1) ...
                * Nitrogen.Temperature_gd.^(Nitrogen.Curve_Fit.alpha_0_gd.coeffs(2)) ...
                + Nitrogen.Curve_Fit.alpha_0_gd.coeffs(3);

    else
        Nitrogen.alpha_0_gd_fit(index1,:) = Nitrogen.alpha_0_gd(index1,:);
    end
```

```
if ((sum(~Nitrogen.Curve_Fit.alpha_bar_gd.nans(index1,:))) >= 20)



    Nitrogen.Curve_Fit.alpha_bar_gd.exclude =  ...
            excludedata( ...
                Nitrogen.Temperature_gd,Nitrogen.alpha_bar_gd(index1,:), ...
                'indices', ...
                ( ...
                    nonzeros( ...
                        (1:size(Nitrogen.Curve_Fit.alpha_bar_gd.nans,2)) ...
                        .*Nitrogen.Curve_Fit.alpha_bar_gd.nans(index1,:) ...
                        ) ...
                )' ...
              );

    Nitrogen.Curve_Fit.alpha_bar_gd.fittype = fittype('fourier6');
    Nitrogen.Curve_Fit.alpha_bar_gd.options = fitoptions('fourier6');
    set( ...
            Nitrogen.Curve_Fit.alpha_bar_gd.options, ...
            'MaxFunEvals',10000,'MaxIter',9000,'TolFun',1e-10,'TolX',1e-10, ...
            'Exclude',Nitrogen.Curve_Fit.alpha_bar_gd.exclude);

    Nitrogen.Curve_Fit.alpha_bar_gd.fit = ...
            fit( ...
                Nitrogen.Temperature_gd', ...
                Nitrogen.alpha_bar_gd(index1,:)', ...
                Nitrogen.Curve_Fit.alpha_bar_gd.fittype, ...
                Nitrogen.Curve_Fit.alpha_bar_gd.options);

    Nitrogen.Curve_Fit.alpha_bar_gd.coeffs = coeffvalues(Nitrogen.Curve_Fit.alpha_bar_gd.fit);


    %smooth the data
```

```
        Nitrogen.alpha_bar_gd_fit(index1,:) = ...
            ( ...
                Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(1) ...
                + Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(2) ...
                    *cos(Nitrogen.Temperature_gd*Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(14)) ...
                + Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(3) ...
                    *sin(Nitrogen.Temperature_gd*Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(14)) ...
                + Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(4) ...
                    *cos(2*Nitrogen.Temperature_gd*Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(14)) ...
                + Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(5) ...
                    *sin(2*Nitrogen.Temperature_gd*Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(14)) ...
                + Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(6) ...
                    *cos(3*Nitrogen.Temperature_gd*Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(14)) ...
                + Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(7) ...
                    *sin(3*Nitrogen.Temperature_gd*Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(14)) ...
                + Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(8) ...
                    *cos(4*Nitrogen.Temperature_gd*Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(14)) ...
                + Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(9) ...
                    *sin(4*Nitrogen.Temperature_gd*Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(14)) ...
                + Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(10) ...
                    *cos(5*Nitrogen.Temperature_gd*Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(14)) ...
                + Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(11) ...
                    *sin(5*Nitrogen.Temperature_gd*Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(14)) ...
                + Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(12) ...
                    *cos(6*Nitrogen.Temperature_gd*Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(14)) ...
                + Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(13) ...
                    *sin(6*Nitrogen.Temperature_gd*Nitrogen.Curve_Fit.alpha_bar_gd.coeffs(14)) ...
            ) ...
            .* ( ...
                (1*~Nitrogen.Curve_Fit.alpha_bar_gd.exclude) ...
                ./(1*~Nitrogen.Curve_Fit.alpha_bar_gd.exclude) ...
            );


    else
        Nitrogen.alpha_bar_gd_fit(index1,:) = Nitrogen.alpha_bar_gd(index1,:);
    end


end
```

```
Nitrogen.alpha_0_gd_fit = (1*~Nitrogen.Curve_Fit.alpha_0_gd.nans) ...
        ./(1*~Nitrogen.Curve_Fit.alpha_0_gd.nans).*Nitrogen.alpha_0_gd_fit;

Nitrogen.alpha_bar_gd_fit = (1*~Nitrogen.Curve_Fit.alpha_bar_gd.nans) ...
        ./(1*~Nitrogen.Curve_Fit.alpha_bar_gd.nans).*Nitrogen.alpha_bar_gd_fit;

%alpha_bar and alpha_0 derivatives with respect to tau
for index1 = 5:length(Nitrogen.Tau_gd)-4
    for index2 = 1:length(Nitrogen.Density_Range)

        Nitrogen.dtau_alpha_bar_gd(index2,index1) = (3*Nitrogen.alpha_bar_gd_fit(index2,index1+4) ...
            - 32*Nitrogen.alpha_bar_gd_fit(index2,index1+3) ...
            + 168*Nitrogen.alpha_bar_gd_fit(index2,index1+2) ...
            - 672*Nitrogen.alpha_bar_gd_fit(index2,index1+1) ...
            + 672*Nitrogen.alpha_bar_gd_fit(index2,index1-1) ...
            - 168*Nitrogen.alpha_bar_gd_fit(index2,index1-2) ...
            + 32*Nitrogen.alpha_bar_gd_fit(index2,index1-3) ...
            - 3*Nitrogen.alpha_bar_gd_fit(index2,index1-4)) ...
            /(-840*Nitrogen.Tau_gd_delta);

        Nitrogen.dtau_alpha_0_gd(index2,index1) = ( 3*Nitrogen.alpha_0_gd_fit(index2,index1+4) ...
            - 32*Nitrogen.alpha_0_gd_fit(index2,index1+3) ...
            + 168*Nitrogen.alpha_0_gd_fit(index2,index1+2) ...
            - 672*Nitrogen.alpha_0_gd_fit(index2,index1+1) ...
            + 672*Nitrogen.alpha_0_gd_fit(index2,index1-1) ...
            - 168*Nitrogen.alpha_0_gd_fit(index2,index1-2) ...
            + 32*Nitrogen.alpha_0_gd_fit(index2,index1-3) ...
            - 3*Nitrogen.alpha_0_gd_fit(index2,index1-4)) ...
            /(-840*Nitrogen.Tau_gd_delta);
    end
end


Nitrogen.Curve_Fit.dtau_alpha_0_gd.nans = isnan(Nitrogen.dtau_alpha_0_gd);
Nitrogen.Curve_Fit.dtau_alpha_bar_gd.nans = isnan(Nitrogen.dtau_alpha_bar_gd);

%Curve fit the derivatives
Nitrogen.Curve_Fit.dtau_alpha_bar_gd.flag = true;
```

```
for index1 = 1:length(Nitrogen.Density_Range)

    if (sum(~Nitrogen.Curve_Fit.dtau_alpha_0_gd.nans(index1,:)) - 4 > 5 )
        Nitrogen.Curve_Fit.dtau_alpha_0_gd.exclude = ...
                excludedata( ...
                    Nitrogen.Temperature_gd(1:length(Nitrogen.Temperature_gd)-4), ...
                    Nitrogen.dtau_alpha_0_gd(index1,:), ...
                    'indices', ...
                    [ ...
                        1:4 ...
                        (nonzeros((1:size(Nitrogen.Curve_Fit.dtau_alpha_0_gd.nans,2))...
                        .*Nitrogen.Curve_Fit.dtau_alpha_0_gd.nans(index1,:)))' ...
                    ] ...
                );

        Nitrogen.Curve_Fit.dtau_alpha_0_gd.fittype = fittype('power2');
        Nitrogen.Curve_Fit.dtau_alpha_0_gd.options = fitoptions('power2');
        set( ...
                Nitrogen.Curve_Fit.dtau_alpha_0_gd.options, ...
                'MaxFunEvals',10000,'MaxIter',9000,'TolFun',1e-10,'TolX',1e-10, ...
                'Exclude',Nitrogen.Curve_Fit.dtau_alpha_0_gd.exclude);

        [Nitrogen.Curve_Fit.dtau_alpha_0_gd.fit ...
            Nitrogen.Curve_Fit.dtau_alpha_0_gd.gof_struct(index1,:)]  ...
                = fit( ...
                    Nitrogen.Temperature_gd(1:length(Nitrogen.Temperature_gd)-4)', ...
                    Nitrogen.dtau_alpha_0_gd(index1,:)', ...
                    Nitrogen.Curve_Fit.dtau_alpha_0_gd.fittype, ...
                    Nitrogen.Curve_Fit.dtau_alpha_0_gd.options);

        Nitrogen.Curve_Fit.dtau_alpha_0_gd.coeffs = coeffvalues(Nitrogen.Curve_Fit.dtau_alpha_0_gd.fit);

        Nitrogen.Curve_Fit.dtau_alpha_0_gd.gof(index1,:) = ...
                [ Nitrogen.Curve_Fit.dtau_alpha_0_gd.gof_struct(index1).rsquare ...
                Nitrogen.Curve_Fit.dtau_alpha_0_gd.gof_struct(index1).sse];

        %smooth the data
        Nitrogen.dtau_alpha_0_gd_fit(index1,:) =  ...
                Nitrogen.Curve_Fit.dtau_alpha_0_gd.coeffs(1) ...
                *Nitrogen.Temperature_gd.^(Nitrogen.Curve_Fit.dtau_alpha_0_gd.coeffs(2)) ...
```

```
            +Nitrogen.Curve_Fit.dtau_alpha_0_gd.coeffs(3);
else
    Nitrogen.dtau_alpha_0_gd_fit(index1,1:length(Nitrogen.Temperature_gd)-4) = ...
            Nitrogen.dtau_alpha_0_gd(index1,:);

    Nitrogen.dtau_alpha_0_gd_fit( ...
            index1, ...
            length(Nitrogen.Temperature_gd)-3:length(Nitrogen.Temperature_gd) ...
            ) = NaN;
end


if(sum(~Nitrogen.Curve_Fit.dtau_alpha_bar_gd.nans(index1,:)) - 4  > 5)

    Nitrogen.Curve_Fit.dtau_alpha_bar_gd.exclude = ...
            excludedata( ...
                Nitrogen.Temperature_gd(1:length(Nitrogen.Temperature_gd)-4), ...
                Nitrogen.dtau_alpha_bar_gd(index1,:), ...
                'indices', ...
                [ ...
                    1:4 ...
                    ( ...
                        nonzeros( ...
                            (1:size(Nitrogen.Curve_Fit.dtau_alpha_bar_gd.nans,2)) ...
                            .*Nitrogen.Curve_Fit.dtau_alpha_bar_gd.nans(index1,:)...
                          ) ...
                     )' ...
                ] ...
              );


    Nitrogen.Curve_Fit.dtau_alpha_bar_gd.fittype = fittype('power2');
    Nitrogen.Curve_Fit.dtau_alpha_bar_gd.options = fitoptions('power2');
    set( ...
            Nitrogen.Curve_Fit.dtau_alpha_bar_gd.options, ...
            'MaxFunEvals',10000,'MaxIter',8000,'TolFun',1e-10,'TolX',1e-10, ...
            'Exclude',Nitrogen.Curve_Fit.dtau_alpha_bar_gd.exclude);
```

```
        [Nitrogen.Curve_Fit.dtau_alpha_bar_gd.fit ...
            Nitrogen.Curve_Fit.dtau_alpha_bar_...
            gd.gof_struct(index1,:)] ...
                = fit( ...
                    Nitrogen.Temperature_gd(1:length(Nitrogen.Temperature_gd)-4)', ...
                    Nitrogen.dtau_alpha_bar_gd(index1,:)', ...
                    Nitrogen.Curve_Fit.dtau_alpha_bar_gd.fittype, ...
                    Nitrogen.Curve_Fit.dtau_alpha_bar_gd.options);

        Nitrogen.Curve_Fit.dtau_alpha_bar_gd.coeffs = ...
                coeffvalues(Nitrogen.Curve_Fit.dtau_alpha_bar_gd.fit);

        Nitrogen.Curve_Fit.dtau_alpha_bar_gd.params(index1,:) =  ...
                [Nitrogen.Curve_Fit.dtau_alpha_bar_gd.gof_struct(index1).rsquare ...
                Nitrogen.Curve_Fit.dtau_alpha_bar_gd.gof_struct(index1).sse ...
                Nitrogen.Curve_Fit.dtau_alpha_bar_gd.coeffs];

        %smooth the data
        Nitrogen.dtau_alpha_bar_gd_fit(index1,:) =  Nitrogen.Curve_Fit.dtau_alpha_bar_gd.coeffs(1)...
                * Nitrogen.Temperature_gd.^(Nitrogen.Curve_Fit.dtau_alpha_bar_gd.coeffs(2)) ...
                + Nitrogen.Curve_Fit.dtau_alpha_bar_gd.coeffs(3);
    else

        Nitrogen.dtau_alpha_bar_gd_fit(index1,1:length(Nitrogen.Temperature_gd)-4) = ...
                Nitrogen.dtau_alpha_bar_gd(index1,:);

        Nitrogen.dtau_alpha_bar_gd_fit( ...
                index1,length(Nitrogen.Temperature_gd)-3:length(Nitrogen.Temperature_gd)) = NaN;
    end

end

Nitrogen.dtau_alpha_0_gd_fit = (1*~Nitrogen.Curve_Fit.alpha_0_gd.nans) ...
        ./(1*~Nitrogen.Curve_Fit.alpha_0_gd.nans).*Nitrogen.dtau_alpha_0_gd_fit;

Nitrogen.dtau_alpha_bar_gd_fit = (1*~Nitrogen.Curve_Fit.alpha_bar_gd.nans) ...
        ./(1*~Nitrogen.Curve_Fit.alpha_bar_gd.nans).*Nitrogen.dtau_alpha_bar_gd_fit;

%Calculate the second derivatives with respect to tau
```

```
for index1 = 5:length(Nitrogen.Tau_gd)-4
    for index2 = 1:length(Nitrogen.Density_Range)

        Nitrogen.d2tau_alpha_bar_gd(index2,index1) = (3*Nitrogen.dtau_alpha_bar_gd_fit(index2,index1+4) ...
            - 32*Nitrogen.dtau_alpha_bar_gd_fit(index2,index1+3) ...
            + 168*Nitrogen.dtau_alpha_bar_gd_fit(index2,index1+2) ...
            - 672*Nitrogen.dtau_alpha_bar_gd_fit(index2,index1+1) ...
            + 672*Nitrogen.dtau_alpha_bar_gd_fit(index2,index1-1) ...
            - 168*Nitrogen.dtau_alpha_bar_gd_fit(index2,index1-2) ...
            + 32*Nitrogen.dtau_alpha_bar_gd_fit(index2,index1-3) ...
            - 3*Nitrogen.dtau_alpha_bar_gd_fit(index2,index1-4)) ...
            /(-840*Nitrogen.Tau_gd_delta);

        Nitrogen.d2tau_alpha_0_gd(index2,index1) = ( 3*Nitrogen.dtau_alpha_0_gd_fit(index2,index1+4) ...
            - 32*Nitrogen.dtau_alpha_0_gd_fit(index2,index1+3) ...
            + 168*Nitrogen.dtau_alpha_0_gd_fit(index2,index1+2) ...
            - 672*Nitrogen.dtau_alpha_0_gd_fit(index2,index1+1) ...
            + 672*Nitrogen.dtau_alpha_0_gd_fit(index2,index1-1) ...
            - 168*Nitrogen.dtau_alpha_0_gd_fit(index2,index1-2) ...
            + 32*Nitrogen.dtau_alpha_0_gd_fit(index2,index1-3) ...
            - 3*Nitrogen.dtau_alpha_0_gd_fit(index2,index1-4)) ...
            /(-840*Nitrogen.Tau_gd_delta);
    end
end


%Calculate Cv
for index1 = 5:length(Nitrogen.Tau_gd)-4
    for index2 = 1:length(Nitrogen.Density_Range)
        Nitrogen.Cv(index2,index1) = -Nitrogen.R*Nitrogen.Tau_gd(index1)^2 ...
        *( Nitrogen.d2tau_alpha_bar_gd(index2,index1) ...
        ...
        + Nitrogen.d2tau_alpha_0_gd(index2,index1));


    end
end

%Calculate S
```

```
for index1 = 5:length(Nitrogen.Tau_gd)-4
    for index2 = 1:length(Nitrogen.Density_Range)
        Nitrogen.S(index2,index1) = Nitrogen.R * ...
            (Nitrogen.T_c/Nitrogen.Temperature_gd(index1) ...
            *(Nitrogen.dtau_alpha_bar_gd_fit(index2,index1) ...
            ...
            + (Nitrogen.dtau_alpha_0_gd_fit(index2,index1))) ...
            - Nitrogen.alpha_bar_gd_fit(index2,index1) ...
            - Nitrogen.alpha_0_gd_fit(index2,index1));

    end
end


%Find good values
Nitrogen.rem_nans = find( ...
        ~isnan( ...
            reshape( ...
                permute( ...
                    Nitrogen.Pressure_gd(1:length(Nitrogen.Density_Range), ...
                    5:length(Nitrogen.Temperature_gd)-4 ...
                    ), ...
                    [2 1] ...
                  ), ...
                1, ...
                numel(Nitrogen.Cv(1:length(Nitrogen.Density_Range),5:length(Nitrogen.Temperature_gd)-4))...
              ) ...
            ) == 1);

%reshape the array to a vector and remove bad values
Nitrogen.Pressure_gd_reshape = ...
        reshape( ...
            permute( ...
                Nitrogen.Pressure_gd(1:length(Nitrogen.Density_Range), ...
                5:length(Nitrogen.Temperature_gd)-4), ...
                [2 1] ...
              ), ...
            1, ...
            numel(Nitrogen.Cv(1:length(Nitrogen.Density_Range),5:length(Nitrogen.Temperature_gd)-4)) ...
          );
```

```
Nitrogen.Pressure_unshape = Nitrogen.Pressure_gd_reshape(Nitrogen.rem_nans);

%reshape the array to a vector and remove bad values
Nitrogen.Temperature_gd_reshape = ...
        reshape( ...
            permute( ...
                Nitrogen.Temperature_gd(ones(1,length(Nitrogen.Density_Range)), ...
                5:length(Nitrogen.Temperature_gd)-4), ...
                [2 1] ...
              ), ...
              1, ...
              numel(Nitrogen.Cv(1:length(Nitrogen.Density_Range),5:length(Nitrogen.Temperature_gd)-4)) ...
            );

Nitrogen.Temperature_unshape = Nitrogen.Temperature_gd_reshape(Nitrogen.rem_nans);

%reshape the array to a vector and remove bad values
Nitrogen.Cv_reshape = ...
        reshape( ...
            permute( ...
                Nitrogen.Cv(1:length(Nitrogen.Density_Range), ...
                5:length(Nitrogen.Temperature_gd)-4), ...
                [2 1] ...
              ), ...
              1, ...
              numel(Nitrogen.Cv(1:length(Nitrogen.Density_Range),5:length(Nitrogen.Temperature_gd)-4)) ...
            );

Nitrogen.Cv_unshape = Nitrogen.Cv_reshape(Nitrogen.rem_nans);

%Change axes to temperature and pressure
Nitrogen.Cv_gd = griddata(Nitrogen.Temperature_unshape, ...
    Nitrogen.Pressure_unshape, ...
    Nitrogen.Cv_unshape,Nitrogen.Temperature,Nitrogen.Pressure','cubic');
```

Appendix O: Sim_Setup_PP.m

```
%% Sim_Setup_PP.m
% This file removes unnecessary fields from the structures. It then saves
% the structures to a file, clears the workspace, and loads the data. This
% is done to prevent memory fragmentation problems.

clear System
Engine = rmfield(Engine, 'BSFC_PQW_Map');
clear Map_Steps
Motor = rmfield(Motor, {'Efficiency', 'Efficiency_Map'});
Pump = rmfield( ...
        Pump, ...
        { 'Efficiency', 'Efficiency_Map', 'Disp_PQW_Map', ...
          'Torque_PQW_Map', 'Efficiency_PQW_Map'});

clear index1 index2 index3

temp.Pressure = Nitrogen.Pressure;
temp.Temperature = Nitrogen.Temperature;
temp.Density = Nitrogen.Density;
temp.Density_Reference = Nitrogen.Density_Reference;
temp.Cv_gd = Nitrogen.Cv_gd;
clear Nitrogen
Nitrogen = temp;
clear temp;

files = dir('.\Data\Data*.mat');
if (~isempty(files))
    save(['.\Data\Data' num2str(length(files)+1) '.mat'])
else
    save('.\Data\Data1.mat')
end

clear all
files = dir('.\Data\Data*.mat');
names = {files.name};
load(char(names(numel(names))));
clear files
clear names
```

Appendix P: Input_Struct.m

```
%% Input_Struct.m
% This file prepares the inputs for the simulation

Inputs = struct;

if (length(Regen_Enable) < Sim_Count)
    Regen_Enable(1:Sim_Count) = Regen_Enable;
end
if (length(Use_Regen_First) < Sim_Count)
    Use_Regen_First(1:Sim_Count) = Use_Regen_First;
end
if (length(Charge_Enable) < Sim_Count)
    Charge_Enable(1:Sim_Count) = Charge_Enable;
end
if (length(Engine_Shutdown) < Sim_Count)
    Engine_Shutdown(1:Sim_Count) = Engine_Shutdown;
end
if (length(dt) < Sim_Count)
    dt(1:Sim_Count) = dt;
end
if (length(Accumulator_Model) < Sim_Count)
    Accumulator_Model(1:Sim_Count) = Accumulator_Model;
end
if (length(Charge_Pressure) < Sim_Count)
    Charge_Pressure(1:Sim_Count) = Charge_Pressure;
end
if (length(Pressure_Buffer) < Sim_Count)
    Pressure_Buffer(1:Sim_Count) = Pressure_Buffer;
end
if (length(Precharge) < Sim_Count)
    Precharge(1:Sim_Count) = Precharge;
end
if (length(Pressure_Init) < Sim_Count)
    Pressure_Init(1:Sim_Count) = Pressure_Init;
end
if (length(Precharge_Temp) < Sim_Count)
    Precharge_Temp(1:Sim_Count) = Precharge_Temp;
end
if (length(Nitrogen_Temp_init) < Sim_Count)
    Nitrogen_Temp_init(1:Sim_Count) = Nitrogen_Temp_init;
end
if (length(Regen_Discharge_Min_Motor_Speed) < Sim_Count)
    Regen_Discharge_Min_Motor_Speed(1:Sim_Count) = ...
            Regen_Discharge_Min_Motor_Speed;
end
if (length(Vehicle_Weight) < Sim_Count)
    Vehicle_Weight(1:Sim_Count) = Vehicle_Weight;
end
if (length(Driving_Cycle) < Sim_Count)
    Driving_Cycle(1:Sim_Count) = Driving_Cycle;
end


for index1 = 1:Sim_Count
```

```
Inputs(index1).Regen_Enable = Regen_Enable(index1);
Inputs(index1).Use_Regen_First = Use_Regen_First(index1);
Inputs(index1).Charge_Enable = Charge_Enable(index1);
Inputs(index1).Engine_Shutdown = Engine_Shutdown(index1);
Inputs(index1).dt = dt(index1);
Inputs(index1).Accumulator_Model = Accumulator_Model(index1);
Inputs(index1).Charge_Pressure = Charge_Pressure(index1);
Inputs(index1).Pressure_Buffer = Pressure_Buffer(index1);
Inputs(index1).Precharge = Precharge(index1);
Inputs(index1).Pressure_Init = Pressure_Init(index1);
Inputs(index1).Precharge_Temp = Precharge_Temp(index1);
Inputs(index1).Nitrogen_Temp_init = Nitrogen_Temp_init(index1);
Inputs(index1).Regen_Discharge_Min_Motor_Speed = ...
        Regen_Discharge_Min_Motor_Speed(index1);
Inputs(index1).Regen_Enable = Regen_Enable(index1);
Inputs(index1).Vehicle_Weight = Vehicle_Weight(index1);

if (strcmpi(Driving_Cycle(index1),'FTP'))
    Inputs(index1).sched_vt = Schedule.FTP.Cycle(:,1);
    Inputs(index1).sched_v = Schedule.FTP.Cycle(:,2);
    Inputs(index1).Driving_Cycle = 'FTP';
elseif (strcmpi(Driving_Cycle(index1),'FTPMOD'))
    Inputs(index1).sched_vt = Schedule.FTP.Cycle(1:767,1);
    Inputs(index1).sched_v = Schedule.FTP.Cycle(1:767,2);
    Inputs(index1).Driving_Cycle = 'FTPMOD';
elseif (strcmpi(Driving_Cycle(index1),'FTPX'))
    Inputs(index1).sched_vt = 0:Cycles*length(Schedule.FTP.Cycle(:,1))-1;
    Inputs(index1).sched_v = reshape( ...
            Schedule.FTP.Cycle(:,ones(1,Cycles)*2), ...
            1, ...
            numel(Schedule.FTP.Cycle(:,2))*Cycles);
    Inputs(index1).Driving_Cycle = {'FTPx' num2str(Cycles)};
elseif (strcmpi(Driving_Cycle(index1),'HWY'))
    Inputs(index1).sched_vt = Schedule.HWYCOL.Cycle(:,1);
    Inputs(index1).sched_v = Schedule.HWYCOL.Cycle(:,2);
    Inputs(index1).Driving_Cycle = 'HWY';
elseif (strcmpi(Driving_Cycle(index1),'HWYX'))
    Inputs(index1).sched_vt = ...
            0:Cycles*length(Schedule.HWYCOL.Cycle(:,1))-1;
    Inputs(index1).sched_v = reshape( ...
            Schedule.HWYCOL.Cycle(:,ones(1,Cycles)*2), ...
            1, ...
            numel(Schedule.HWYCOL.Cycle(:,2))*Cycles);
    Inputs(index1).Driving_Cycle = {'HWYx' num2str(Cycles)};
elseif (strcmpi(Driving_Cycle(index1),'HWYMOD'))
    Inputs(index1).sched_vt = Schedule.HWYCOL.Cycle(1:60,1);
    Inputs(index1).sched_v = Schedule.HWYCOL.Cycle(1:60,2);
    Inputs(index1).Driving_Cycle = 'HWYMOD';
elseif (strcmpi(Driving_Cycle(index1),'SC03'))
    Inputs(index1).sched_vt = Schedule.SC03.Cycle(:,1);
    Inputs(index1).sched_v = Schedule.SC03.Cycle(:,2);
    Inputs(index1).Driving_Cycle = 'SC03';
elseif (strcmpi(Driving_Cycle(index1),'SC03X'))
    Inputs(index1).sched_vt = ...
            0:Cycles*length(Schedule.SC03.Cycle(:,1))-1;
    Inputs(index1).sched_v = reshape( ...
            Schedule.SC03.Cycle(:,ones(1,Cycles)*2), ...
```

```
                1, ...
                numel(Schedule.SC03.Cycle(:,2))*Cycles);
        Inputs(index1).Driving_Cycle = {'SC03x' num2str(Cycles)};
    elseif (strcmpi(Driving_Cycle(index1),'US06'))
        Inputs(index1).sched_vt = Schedule.US06.Cycle(:,1);
        Inputs(index1).sched_v = Schedule.US06.Cycle(:,2);
        Inputs(index1).Driving_Cycle = 'US06';
    end

end

%cleanup
clear Regen_Enable Use_Regen_First Charge_Enable Engine_Shutdown dt ...
    Accumulator_Model Charge_Pressure Pressure_Buffer Precharge ...
    Pressure_Init Precharge_Temp Nitrogen_Temp_init ...
    Regen_Discharge_Min_Motor_Speed Driving_Cycle
```

Appendix Q: Multi_Sim_PP

```
%% Multi_Sim_PP
% This function handles passing simulation settings and system data to the
% simulation function


function Sim_Results = Multi_Sim_PP( ...
        Simulation, Hybrid, Hydrostatic, Hydraulics, Motor, Pump, ...
        Engine, Vehicle, Environment, BWR, Nitrogen, Foam, Inputs, ...
        Energy_Balance)

    %Assign inputs to their proper structures and fields
    Simulation.Regen_Enable = Inputs.Regen_Enable;
    Simulation.Use_Regen_First = Inputs.Use_Regen_First;
    Simulation.Charge_Enable = Inputs.Charge_Enable;
    Simulation.Engine_Shutdown = Inputs.Engine_Shutdown;
    Simulation.dt = Inputs.dt;
    Simulation.Accumulator_Model = Inputs.Accumulator_Model;
    Simulation.Charge_Pressure = Inputs.Charge_Pressure;
    Simulation.Pressure_Buffer = Inputs.Pressure_Buffer;
    Hydraulics.Pressure_Init = Inputs.Pressure_Init;
    Hydraulics.Precharge = Inputs.Precharge;
    Hydraulics.Precharge_Temp = Inputs.Precharge_Temp;
    Hydraulics.Nitrogen_Temp_init = Inputs.Nitrogen_Temp_init;
    Simulation.Regen_Discharge_Min_Motor_Speed = ...
            Inputs.Regen_Discharge_Min_Motor_Speed;
    Vehicle.Weight = Inputs.Vehicle_Weight;
    Simulation.sched_vt = Inputs.sched_vt;
    Simulation.sched_v = Inputs.sched_v;



    Simulation.Inputs = Inputs;
    Simulation.Inputs.Pump_Displacement_Max = Pump.Displacement_Max;
    Simulation.Inputs.Motor_Displacement_Max = Motor.Displacement_Max;
    Simulation.Inputs.Driving_Cycle = Inputs.Driving_Cycle;
    Simulation.Inputs = rmfield(Simulation.Inputs, {'sched_vt','sched_v'});

    %If energy balance is set, the end state of the energy storage system
    %is used as in initial condition for a second run on the simulation
    if (Energy_Balance)
        for index = 1:2
            if(index == 2)
                Hydraulics.Nitrogen_Temp_init = ...
                        Sim_Results.Nitrogen_Temperature( ...
                            length(Sim_Results.Nitrogen_Temperature) ...
                            );
                Hydraulics.Pressure_Init = ...
                        Sim_Results.Pressure(length(Sim_Results.Pressure));

                Simulation.Energy_Pumped_Init = ...
                        Sim_Results.Energy_Pumped( ...
                            length(Sim_Results.Energy_Pumped));

                Simulation.BSFC_Energy_Total_Init = ...
                        Sim_Results.BSFC_Energy_Total( ...
```

```
                        length(Sim_Results.BSFC_Energy_Total));

            Simulation.Regen_Energy_Init = ...
                    Sim_Results.Regen_Energy( ...
                        length(Sim_Results.Regen_Energy));

            clear Sim_Results
        end


        Sim_Results = Run_Sim_PP( ...
                Simulation, Hybrid, Hydrostatic, Hydraulics, Motor, ...
                Pump, Engine, Vehicle, Environment, BWR, Nitrogen, Foam);

    end
else
    Sim_Results = Run_Sim_PP( ...
            Simulation, Hybrid, Hydrostatic, Hydraulics, Motor, Pump, ...
            Engine, Vehicle, Environment, BWR, Nitrogen, Foam);
end
```

Appendix R: RunSim_PP.m

```
%% Run_Sim_PP
% Functionalized version of Run_Sim to allow Parallel Processing
% This function runs the simulation
% A best-efficiency control strategy is used
%
% File Structure:
%   -Initialize variables
%   -Set initial conditions
%   -Begin Simulation
%       -Calculate HP Accumulator volume
%       -Calculate system pressure
%       -Calculate driving cycle velocity at next timestep
%       -Calculate required acceleration
%       -Calculate average BSFC of stored energy
%       -Calculate amount of energy stored from regen
%       -Determine motor operation
%       -Determine engine/pump operation
%       -Calculate mileage
%   -End of Simulation
%   -Cleanup output structure
%
%
% File Dependencies
%   -Hydrostatic_Drive
%   -Accumulator_Drive
%   -Interp2_Reduction
%   -Interp3_Reduction
%
%Input Structure List with required fields
%   Simulation                              %   Hydrostatic
%       -dt                                 %       -Effective_BSFC
%       -sched_v                            %       -Engine_Speed
%       -sched_vt                           %       -Engine_Torque
%       -Regen_Enable                       %       -Engine_BSFC
%       -Charge_Enable                      %       -Pump_Disp
%       -Use_Regen_First                    %       -Pump_Vol_Efficiency
%       -continue                           %       -Pump_Mech_Efficiency
%       -                                   %       -Pump_Efficiency
Regen_Discharge_Min_Motor_Speed             %       -Motor_Vol_Efficiency
%       -Charge_Pressure                    %       -Motor_Mech_Efficiency
%       -Buffer_Pressure                    %       -Motor_Efficiency
%       -Energy_Pumped_Init                 %       -Motor_Displacement
%       -BSFC_Energy_Total_Init             %       -Motor_Flow_Rate
%       -Accumulator_Model                  %       -Max_Torque
%   Hybrid                                  %       -MPG
%       -Engine_BSFC                        %       -Pressure
%       -Engine_Speed                       %   Hydraulics
%       -Engine_Torque                      %       -Precharge
%       -Effective_BSFC                     %       -Pressure_Init
%       -Pump_Speed                         %       -HP_ACC_Vol_Init
%       -Pump_Torque                        %       -HP_ACC_Gas_Vol_Max
%       -Pump_Displacement                  %       -Precharge_Temp
%       -Pump_Vol_Efficiency                %       -Nitrogen_Temp_init
%       -Pump_Mech_Efficiency               %   Motor
%       -Pump_Efficiency                    %       -Pressure_Range
```

```
%       -Speed_Range
%       -Displacement_Range
%       -Displacement_Max
%       -Flow_Rate_Map
%       -Displacement_Map
%       -Vol_Efficiency_Map
%       -Mech_Efficiency_Map
%   Pump
%       -Pressure_Range
%       -Speed_Range
%       -Torque_Range
%       -Displacement_Range
%       -Displacement_Max
%       -Displacement_Map
%       -Vol_Efficiency_Map
%       -Mech_Efficiency_Map
%   Engine
%       -Speed_Range
%       -Torque_Range
%       -FFR_Idle
%       -BSFC_Map
%   Vehicle
%       -RL
%           A
%           B
%           C
%       -Differential_Ratio
%       -Tire_Diameter
%       -Weight
%       -CdA
%       -Gearbox_Efficiency
%       -Wheel_MOI
%       -EP_Gear_Ratios
%       -Motor_Gear_Ratios
%   Environment
%       -Air_Density
%       -Diesel_Density
%   BWR
%       -a
%       -A_0
%       -b
%       -B_0
%       -c
%       -C_0
%       -alpha
%       -gamma
%       -R
%   Nitrogen
%       -Pressure
%       -Temperature
%       -Density
%       -Density_Reference
%       -Cv_gd
%   Foam
%       -mass
%       -c
%
%
%
%
%%Simout Fields
% -Settings
%   dt
%   Inputs
%   Interp2_Range
%   Interp3_Range
%   Regen_Enable
%   Use_Regen_First
%   Charge_Enable
%   Engine_Shutdown
%   Accumulator_Model
%   Charge_Pressure
%   Pressure_Buffer
%   Regen_Discharge_Min_Motor_Speed
%
% -Operation modes
%   Hydrostatic
%   regen
%   idle
%   charge
%   Mechanical_Brakes
%   idle_fuel
%   Over_Pressure
%
% -Initial Conditions
%   Energy_Pumped_Init
%   BSFC_Energy_Total_Init
%   Regen_Energy_Init
%   DataFile
%
% -Motor
%   Motor_Displacement
%   Motor_Speed
%   Motor_Torque
%   Motor_Vol_Efficiency
%   Motor_Mech_Efficiency
%   Motor_Flow_Rate
%
% -Pump
%   Pump_Displacement
%   Pump_Speed
%   Pump_Vol_Efficiency
%   Pump_Mech_Efficiency
%   Pump_Flow_Rate
%
% -Engine
%   Engine_Speed
```

```
%   Engine_Torque
%   Engine_BSFC
%
% -Nitrogen
%   Nitrogen_Temperature
%   Nitrogen_Specific_Volume
%   Nitrogen_Cv
%
% -Energy Storage
%   BSFC_Vol_Total
%   Fluid_Vol_Pumped
%   BSFC_AVG
%   Pressure
%   HP_ACC_Vol
%   Regen_Energy
%   Energy_Pumped
%   BSFC_Energy_Total
%
% -Driving Cycle
%   sched_v
%   sched_vt
%   tspan
%   steps
%   t
%   Position
%   Velocity
%   Acceleration
%
% -Fuel Efficiency
%   Fuel_Mass
%   Fuel_Mass_Total
%   MPG_Current
%   MPG_Average
%   Alt_Effective_BSFC
%
% -Other
%   EP_Gear
%   Motor_Gear
%   Hydrostatic_Pressure
```

```
function simout = Run_Sim_PP(Simulation, Hybrid, Hydrostatic, Hydraulics, Motor, Pump, Engine, Vehicle,
Environment, BWR, Nitrogen, Foam)

    if (~Simulation.continue)
    Simulation.BSFC_Vol_Total = 0;
    Simulation.Fluid_Vol_Pumped = 0;
    Simulation.BSFC_AVG = 0;

    %The display counter is used in the non-functionalized Run_Sim to
    %display data during the simulation
    %display_counter = 1;

    Simulation.tspan(1) = min(Simulation.sched_vt);
    Simulation.tspan(2) = max(Simulation.sched_vt);

    %Initialize Vars
    Simulation.steps = 1:(Simulation.tspan(2)-Simulation.tspan(1))/Simulation.dt;
    Simulation.t(Simulation.steps) = 0;
    Simulation.Position(Simulation.steps) = 0;
    Simulation.Velocity(Simulation.steps) = 0;
    Simulation.Acceleration(Simulation.steps) = 0;
    Simulation.Pressure(Simulation.steps) = 0;
    Simulation.HP_ACC_Vol(Simulation.steps) = 0;
    Simulation.Motor_Displacement(Simulation.steps) = 0;
    Simulation.Motor_Speed(Simulation.steps) = 0;
    Simulation.Motor_Torque(Simulation.steps) = 0;
    Simulation.Motor_Vol_Efficiency(Simulation.steps) = 0;
    Simulation.Motor_Mech_Efficiency(Simulation.steps) = 0;
    Simulation.Motor_Flow_Rate(Simulation.steps) = 0;
    Simulation.Pump_Displacement(Simulation.steps) = 0;
    Simulation.Pump_Speed(Simulation.steps) = 0;
    Simulation.Pump_Vol_Efficiency(Simulation.steps) = 0;
    Simulation.Pump_Mech_Efficiency(Simulation.steps) = 0;
    Simulation.Pump_Flow_Rate(Simulation.steps) = 0;
    Simulation.Engine_Speed(Simulation.steps) = 0;
    Simulation.Engine_Torque(Simulation.steps) = 0;
    Simulation.Engine_BSFC(Simulation.steps) = 0;
    Simulation.Fuel_Mass(Simulation.steps) = 0;
    Simulation.Fuel_Mass_Total(Simulation.steps) = 0;
    Simulation.MPG_Current(Simulation.steps) = 0;
```

```
Simulation.MPG_Average(Simulation.steps) = 0;
Simulation.EP_Gear(Simulation.steps) = 0;
Simulation.Motor_Gear(Simulation.steps) = 0;
Simulation.Hydrostatic_Pressure(Simulation.steps) = 0;
Simulation.Regen_Energy(Simulation.steps) = 0;
Simulation.Alt_Effective_BSFC(Simulation.steps) = 0;
Simulation.Nitrogen_Temperature(Simulation.steps) = 0;
Simulation.Nitrogen_Specific_Volume(Simulation.steps) = 0;
Simulation.Nitrogen_Cv(Simulation.steps) = 0;
Simulation.Fluid_Vol_Pumped(Simulation.steps) = 0;
Simulation.BSFC_Vol_Total(Simulation.steps) = 0;
Simulation.BSFC_AVG(Simulation.steps) = 0;
Simulation.Energy_Pumped(Simulation.steps) = 0;
Simulation.BSFC_Energy_Total(Simulation.steps) = 0;

%Operation Modes
Simulation.Hydrostatic(Simulation.steps) = false;
Simulation.regen(Simulation.steps) = false;
Simulation.idle(Simulation.steps) = false;
Simulation.charge(Simulation.steps) = false;
Simulation.Mechanical_Brakes(Simulation.steps) = false;

%Initial Conditions
Simulation.Fluid_Vol_Pumped(1) = Simulation.HP_ACC_Vol(1);
Simulation.Energy_Pumped(1) = Simulation.Energy_Pumped_Init;
Simulation.BSFC_Energy_Total(1) = Simulation.BSFC_Energy_Total_Init;
Simulation.Regen_Energy(1) = Simulation.Regen_Energy_Init;
Simulation.t(1) = Simulation.tspan(1);
Simulation.Velocity(1) = interp1(Simulation.sched_vt,Simulation.sched_v,Simulation.t(1)) ...
    * 1.4666666666666666;
Simulation.Fuel_Mass_Total(1) = 0;
Simulation.idle(1) = true;
Simulation.EP_Gear(1) = 1;
Simulation.Motor_Gear(1) = 1;
Simulation.idle_fuel = 0;
Simulation.Pressure_Low = false;
Simulation.empty = true;

%Accumulator Model
if (strcmpi(Simulation.Accumulator_Model,'BWR'))
```

```
Nitrogen.Specific_Vol_precharge = 1/(Nitrogen.Density_Reference(1) ...
        *interp2_mod( ...
            Nitrogen.Temperature, ...
            Nitrogen.Pressure, ...
            Nitrogen.Density, ...
            Hydraulics.Precharge_Temp, ...
            Hydraulics.Precharge/14.696));
Nitrogen.mass = Hydraulics.HP_ACC_Gas_Vol_Max/Nitrogen.Specific_Vol_precharge;
Nitrogen.Specific_Vol_init = 1/(Nitrogen.Density_Reference(1) ...
        *interp2_mod( ...
        Nitrogen.Temperature, ...
        Nitrogen.Pressure, ...
        Nitrogen.Density, ...
        Hydraulics.Nitrogen_Temp_init, ...
        Hydraulics.Pressure_Init/14.696));
Hydraulics.HP_ACC_Vol_init = (Hydraulics.HP_ACC_Gas_Vol_Max  ...
        - Nitrogen.Specific_Vol_init*Nitrogen.mass) ...
        * 1e6/2.54^3;

Simulation.Nitrogen_Temperature(1) = Hydraulics.Nitrogen_Temp_init;
Simulation.Nitrogen_Specific_Volume(1) = Nitrogen.Specific_Vol_init;
Simulation.HP_ACC_Vol(1) = Hydraulics.HP_ACC_Vol_init;

Simulation.Pressure(1) = ...
        ( ...
        BWR.R * Simulation.Nitrogen_Temperature(1) / Simulation.Nitrogen_Specific_Volume(1) ...
        ...
        +   ( ...
            BWR.B_0 * BWR.R * Simulation.Nitrogen_Temperature(1) - BWR.A_0 ...
            - BWR.C_0 / Simulation.Nitrogen_Temperature(1)^2 ...
            ) ...
        / Simulation.Nitrogen_Specific_Volume(1)^2 ...
        ...
        + (BWR.b * BWR.R * Simulation.Nitrogen_Temperature(1) - BWR.a) ...
        / Simulation.Nitrogen_Specific_Volume(1)^3 ...
        ...
        + BWR.a * BWR.alpha / Simulation.Nitrogen_Specific_Volume(1)^6 ...
        ...
        +   ( ...
            BWR.c * (1 + BWR.gamma / Simulation.Nitrogen_Specific_Volume(1)^2) ...
```

```
            *exp(-BWR.gamma / Simulation.Nitrogen_Specific_Volume(1)^2) ...
            ) ...
        / (Simulation.Nitrogen_Specific_Volume(1)^3 * Simulation.Nitrogen_Temperature(1)^2) ...
        ) * 14.696/101325;

Simulation.Nitrogen_Cv(1) = interp2_mod(Nitrogen.Temperature,Nitrogen.Pressure, ...
        Nitrogen.Cv_gd,Simulation.Nitrogen_Temperature(1),Simulation.Pressure(1)/14.696);

elseif (strcmpi(Simulation.Accumulator_Model,'isothermal'))

    Simulation.Pressure(1) = Hydraulics.Pressure_Init;
    Simulation.HP_ACC_Vol(1) = Hydraulics.HP_ACC_Vol_Max ...
            * (1-Hydraulics.Precharge/Simulation.Pressure(1));

end

    timestep = 2;
end


%% Begin Simulation
for timestep = timestep:(Simulation.tspan(2)-Simulation.tspan(1))/Simulation.dt

prevstep = timestep-1;
Simulation.t(timestep)=Simulation.t(prevstep)+Simulation.dt;

%% Accumulator
Simulation.Over_Pressure = false;
%High Pressure accumulator volume.
if(~Simulation.Hydrostatic(prevstep))
    Simulation.HP_ACC_Vol(timestep) = Simulation.HP_ACC_Vol(prevstep) ...
            + (Simulation.Pump_Flow_Rate(prevstep) - Simulation.Motor_Flow_Rate(prevstep)) ...
            * Simulation.dt;
else
    Simulation.HP_ACC_Vol(timestep) = Simulation.HP_ACC_Vol(prevstep);
end
%Check to see if the High Pressure accumulator contains fluid.
if (Simulation.HP_ACC_Vol(timestep) <= 0)
    Simulation.HP_ACC_Vol(timestep) = 0;
```

```
    Simulation.empty = true;
else
    Simulation.empty = false;
end


%System pressure calculations
%strcmpi is used to check which model is to be used
if (strcmpi(Simulation.Accumulator_Model,'BWR'))
    %Benedict-Webb-Rubin model

    Simulation.Nitrogen_Specific_Volume(timestep) = ...
            (Hydraulics.HP_ACC_Gas_Vol_Max - Simulation.HP_ACC_Vol(timestep)*2.54^3/1e6) ...
            / Nitrogen.mass;

    Simulation.Nitrogen_Temperature(timestep) = ...
            Simulation.Nitrogen_Temperature(prevstep) ...
            ...
            -   ( ...
                BWR.R * Simulation.Nitrogen_Temperature(prevstep) ...
                / Simulation.Nitrogen_Specific_Volume(prevstep) ...
                * (1+BWR.b/Simulation.Nitrogen_Specific_Volume(prevstep)^2) ...
                ...
                +   ( ...
                    BWR.B_0*BWR.R*Simulation.Nitrogen_Temperature(prevstep) ...
                    + 2*BWR.C_0/Simulation.Nitrogen_Temperature(prevstep)^2 ...
                    ) ...
                / Simulation.Nitrogen_Specific_Volume(prevstep)^2 ...
                ...
                - 2*BWR.c * (1 + BWR.gamma/Simulation.Nitrogen_Specific_Volume(prevstep)^2) ...
                * exp(-BWR.gamma/Simulation.Nitrogen_Specific_Volume(prevstep)^2) ...
                /   ( ...
                    Simulation.Nitrogen_Specific_Volume(prevstep)^3 ...
                    * Simulation.Nitrogen_Temperature(prevstep)^2 ...
                    ) ...
                ) ...
            /   ( ...
                Simulation.Nitrogen_Cv(prevstep)/28.0134 ...
                * (1 + Foam.mass*Foam.c /(Nitrogen.mass*Simulation.Nitrogen_Cv(prevstep)/28.0134)) ...
                ) ...
```

```
        *   ( ...
            Simulation.Nitrogen_Specific_Volume(timestep) ...
            - Simulation.Nitrogen_Specific_Volume(prevstep) ...
            );

%Latex Equation
%\[{{P}_{g}}=\frac{RT}{\nu }+\left(
%{{B}_{0}}RT-{{A}_{0}}-{{C}_{0}}/{{T}^{2}} \right)/{{\nu }^{2}}\]

Simulation.Pressure(timestep) = ...
        ( ...
        BWR.R * Simulation.Nitrogen_Temperature(timestep) ...
        /Simulation.Nitrogen_Specific_Volume(timestep) ...
        +   ( ...
            BWR.B_0 * BWR.R*Simulation.Nitrogen_Temperature(timestep) ...
            - BWR.A_0 - BWR.C_0/Simulation.Nitrogen_Temperature(timestep)^2 ...
            ) ...
        / Simulation.Nitrogen_Specific_Volume(timestep)^2 ...
        + (BWR.b*BWR.R*Simulation.Nitrogen_Temperature(timestep) - BWR.a) ...
        / Simulation.Nitrogen_Specific_Volume(timestep)^3 ...
        + BWR.a*BWR.alpha/Simulation.Nitrogen_Specific_Volume(timestep)^6 ...
        +   ( ...
            BWR.c ...
            *   ( ...
                1 + BWR.gamma / Simulation.Nitrogen_Specific_Volume(timestep)^2 ...
                ) ...
            *exp( ...
                -BWR.gamma/Simulation.Nitrogen_Specific_Volume(timestep)^2 ...
                ) ...
            ) ...
        /   ( ...
            Simulation.Nitrogen_Specific_Volume(timestep)^3 ...
            *Simulation.Nitrogen_Temperature(timestep)^2 ...
            ) ...
        ) ...
        *14.696/101325;


Simulation.Temperature_Indices = [ ];
Simulation.Temperature_Indices = Interp2_Reduction(Nitrogen.Temperature, ...
```

```
        Simulation.Nitrogen_Temperature(timestep),Simulation.Interp2_Range);

Simulation.Nitrogen_Cv(timestep) = interp2_mod( ...
        Nitrogen.Temperature(Simulation.Temperature_Indices), ...
        Nitrogen.Pressure,Nitrogen.Cv_gd(:,Simulation.Temperature_Indices), ...
        Simulation.Nitrogen_Temperature(timestep), ...
        Simulation.Pressure(timestep)/14.696);

while (Simulation.Pressure(timestep) > max(Motor.Pressure_Range))
    %Check to see if system pressure is greater than the systems rated
    %pressure. If greater than the rated pressure, fluid is purged (i.e.
    %pressure relief valve)
    Simulation.Over_Pressure = true;
    Simulation.HP_ACC_Vol(timestep) = Simulation.HP_ACC_Vol(timestep) - 0.5;

    Simulation.Nitrogen_Specific_Volume(timestep) = ...
            (Hydraulics.HP_ACC_Gas_Vol_Max - Simulation.HP_ACC_Vol(timestep)*2.54^3/1e6) ...
            /Nitrogen.mass;

    Simulation.Nitrogen_Temperature(timestep) = ...
            Simulation.Nitrogen_Temperature(prevstep) ...
            ...
            -   ( ...
                BWR.R * Simulation.Nitrogen_Temperature(prevstep) ...
                / Simulation.Nitrogen_Specific_Volume(prevstep) ...
                * (1+BWR.b/Simulation.Nitrogen_Specific_Volume(prevstep)^2) ...
                ...
                +   ( ...
                    BWR.B_0*BWR.R*Simulation.Nitrogen_Temperature(prevstep) ...
                    + 2*BWR.C_0/Simulation.Nitrogen_Temperature(prevstep)^2 ...
                    ) ...
                / Simulation.Nitrogen_Specific_Volume(prevstep)^2 ...
                ...
                - 2*BWR.c * (1 + BWR.gamma/Simulation.Nitrogen_Specific_Volume(prevstep)^2) ...
                * exp(-BWR.gamma/Simulation.Nitrogen_Specific_Volume(prevstep)^2) ...
                /   ( ...
                    Simulation.Nitrogen_Specific_Volume(prevstep)^3 ...
                    * Simulation.Nitrogen_Temperature(prevstep)^2 ...
                    ) ...
                ) ...
```

```
                    /   ( ...
                        Simulation.Nitrogen_Cv(prevstep)/28.0134 ...
                        * (1 + Foam.mass*Foam.c /(Nitrogen.mass*Simulation.Nitrogen_Cv(prevstep)/28.0134))
...
                        ) ...
                    *   ( ...
                        Simulation.Nitrogen_Specific_Volume(timestep) ...
                        - Simulation.Nitrogen_Specific_Volume(prevstep) ...
                        );



        Simulation.Pressure(timestep) = ...
                    ( ...
                    BWR.R * Simulation.Nitrogen_Temperature(timestep) ...
                    / Simulation.Nitrogen_Specific_Volume(timestep) ...
                    ...
                    +   ( ...
                        BWR.B_0 * BWR.R * Simulation.Nitrogen_Temperature(timestep) - BWR.A_0 ...
                        - BWR.C_0/Simulation.Nitrogen_Temperature(timestep)^2 ...
                        ) ...
                    / Simulation.Nitrogen_Specific_Volume(timestep)^2 ...
                    ...
                    + (BWR.b * BWR.R * Simulation.Nitrogen_Temperature(timestep) - BWR.a) ...
                    / Simulation.Nitrogen_Specific_Volume(timestep)^3 ...
                    ...
                    + BWR.a*BWR.alpha / Simulation.Nitrogen_Specific_Volume(timestep)^6 ...
                    ...
                    +   ( ...
                        BWR.c * (1 + BWR.gamma/Simulation.Nitrogen_Specific_Volume(timestep)^2) ...
                        * exp(-BWR.gamma/Simulation.Nitrogen_Specific_Volume(timestep)^2) ...
                        ) ...
                    /   ( ...
                        Simulation.Nitrogen_Specific_Volume(timestep)^3 ...
                        *Simulation.Nitrogen_Temperature(timestep)^2 ...
                        ) ...
                    ...
                    ) ...
                    *14.696/101325;
            end
```

```
elseif (strcmpi(Simulation.Accumulator_Model,'isothermal'))
    %Isothermal model
    %System pressure is calculated using the High Pressure accumulator
    %volume and precharge pressure
    Simulation.Pressure(timestep) =  Hydraulics.Precharge*Hydraulics.HP_ACC_Vol_Max ...
            /(Hydraulics.HP_ACC_Vol_Max - Simulation.HP_ACC_Vol(timestep));

    %Check to see if system pressure is greater than the systems rated
    %pressure. If greater than the rated pressure, fluid is purged (i.e.
    %pressure relief valve)
    if (Simulation.Pressure(timestep) > max(Motor.Pressure_Range))
        Simulation.Pressure(timestep) = max(Motor.Pressure_Range);
        Simulation.HP_ACC_Vol(timestep) = Hydraulics.HP_ACC_Vol_Max ...
                - Hydraulics.Precharge * Hydraulics.HP_ACC_Vol_Max ...
                / Simulation.Pressure(timestep);
    end
end


%% Velocity (ft/sec)
Simulation.Velocity(timestep) = Simulation.Velocity(prevstep) ...
        + Simulation.Acceleration(prevstep)*Simulation.dt;
if(Simulation.Velocity(timestep) < 0 )
    Simulation.Velocity(timestep) = 0;
end

%Differential and motor gearbox output shaft speed
Simulation.Differential_Speed = Simulation.Velocity(timestep) ...
        /(pi*Vehicle.Tire_Diameter)*60*Vehicle.Differential_Ratio;


%EPA driving cycles are listed in MPH, convert to ft/sec for
%calculations
Simulation.acc = ...
        ( ...
        interp1(Simulation.sched_vt, Simulation.sched_v,Simulation.t(timestep)+Simulation.dt) ...
        *5280/3600-Simulation.Velocity(timestep) ...
        ) ...
        /Simulation.dt;
```

```
%% Stored energy BSFC
%Keeps track of the energy averaged EP BSFC of the fluid in the HP
%Accumulator. Used when determining the most efficient mode of
%operation. BSFC during regen is treated as 0.
%
%When the net flow into the HP accumulator is positive and the system
%is not in Hydrostatic mode:
if ( ...
        ( ...
        (Simulation.Pump_Flow_Rate(prevstep) - Simulation.Motor_Flow_Rate(prevstep)) ...
        *Simulation.dt > 0 ...
        ) ...
    && ~Simulation.Hydrostatic(prevstep) ...
    )


    %Check to see if regen was enabled on the last timestep. Energy
    %reclaimed using regen does not count towards the energy totals
    if (Simulation.charge(prevstep) && ~Simulation.regen(prevstep))
        Simulation.Energy_Pumped(timestep) = Simulation.Energy_Pumped(prevstep) ...
                + (Simulation.Pump_Flow_Rate(prevstep) - Simulation.Motor_Flow_Rate(prevstep)) ...
                *Simulation.Pressure(prevstep)*Simulation.dt;

        Simulation.BSFC_Energy_Total(timestep) = ...
                Simulation.BSFC_Energy_Total(prevstep) ...
                ...
                + Simulation.Engine_BSFC(prevstep) ...
                /   ( ...
                    Simulation.Pump_Vol_Efficiency(prevstep) ...
                    * Simulation.Pump_Mech_Efficiency(prevstep) ...
                    * Vehicle.Gearbox_Efficiency ...
                    ) ...
                * Simulation.Pump_Flow_Rate(prevstep) ...
                * Simulation.Pressure(prevstep) * Simulation.dt  ...
                ...
                - Simulation.BSFC_AVG(prevstep) ...
                * Simulation.Motor_Flow_Rate(prevstep) ...
                * Simulation.Pressure(prevstep)*Simulation.dt;
```

```
elseif (Simulation.regen(prevstep))
    Simulation.Energy_Pumped(timestep) = Simulation.Energy_Pumped(prevstep) ...
            + Simulation.Pump_Flow_Rate(prevstep) ...
            * Simulation.Pressure(prevstep) * Simulation.dt;

    %Only fluid added to the HP Accumulator by the Engine/Pump
    %adds to the BSFC_Vol_Total
    %Commented code at EOL will include regen.
    if (~Simulation.idle(prevstep))
        Simulation.BSFC_Energy_Total(timestep) = ...
                Simulation.BSFC_Energy_Total(prevstep) ...
                ...
                + Simulation.Engine_BSFC(prevstep) ...
                /   ( ...
                    Simulation.Pump_Vol_Efficiency(prevstep) ...
                    * Simulation.Pump_Mech_Efficiency(prevstep) ...
                    * Vehicle.Gearbox_Efficiency) ...
                    * (Simulation.Pump_Flow_Rate(prevstep)) ...
                    * Simulation.Pressure(prevstep)*Simulation.dt;
                    % - Simulation.BSFC_AVG(prevstep) ...
                    % * Simulation.Motor_Flow_Rate(prevstep) ...
                    % * Simulation.Pressure(prevstep)*Simulation.dt;
    else
        Simulation.BSFC_Energy_Total(timestep) = Simulation.BSFC_Energy_Total(prevstep);
    end
end


%If the energy pumped or energy total becomes zero or less than
%zero, the values are reset to zero and the BSFC average is set to
%the value from the previous timestep.
if (Simulation.BSFC_Energy_Total(timestep) <= 0)
    Simulation.BSFC_Energy_Total(timestep) = 0;
    Simulation.Energy_Pumped(timestep) = 0;
end

if (Simulation.Energy_Pumped(timestep) <= 0)
    Simulation.Energy_Pumped(timestep) = 0;
```

```
        Simulation.BSFC_AVG(timestep) = Simulation.BSFC_AVG(prevstep);
    else
        Simulation.BSFC_AVG(timestep) = Simulation.BSFC_Energy_Total(timestep) ...
                / Simulation.Energy_Pumped(timestep);
    end


%When the net flow into the HP accumulator is negative and the system
%is not in Hydrostatic mode:
elseif ( ...
        ( ...
        (Simulation.Pump_Flow_Rate(prevstep) - Simulation.Motor_Flow_Rate(prevstep)) ...
        * Simulation.dt < 0 ...
        ) ...
        && ~Simulation.Hydrostatic(prevstep) ...
        )

    Simulation.Energy_Pumped(timestep) = Simulation.Energy_Pumped(prevstep) ...
            + (Simulation.Pump_Flow_Rate(prevstep) - Simulation.Motor_Flow_Rate(prevstep)) ...
            * Simulation.Pressure(prevstep)*Simulation.dt;

    if (~Simulation.idle(prevstep))
        Simulation.BSFC_Energy_Total(timestep) = ...
                Simulation.BSFC_Energy_Total(prevstep) ...
                ...
                + Simulation.Engine_BSFC(prevstep) ...
                /   ( ...
                    Simulation.Pump_Vol_Efficiency(prevstep) ...
                    * Simulation.Pump_Mech_Efficiency(prevstep) ...
                    *Vehicle.Gearbox_Efficiency ...
                    ) ...
                * Simulation.Pump_Flow_Rate(prevstep) ...
                * Simulation.Pressure(prevstep)* Simulation.dt ...
                - Simulation.BSFC_AVG(prevstep) ...
                * Simulation.Motor_Flow_Rate(prevstep) ...
                * Simulation.Pressure(prevstep) * Simulation.dt;
    else
        Simulation.BSFC_Energy_Total(timestep) = ...
                Simulation.BSFC_Energy_Total(prevstep) ...
                - Simulation.BSFC_AVG(prevstep) ...
```

```
                * Simulation.Motor_Flow_Rate(prevstep) ...
                * Simulation.Pressure(prevstep) * Simulation.dt;
    end


    %If the energy pumped or energy total becomes zero or less than
    %zero, the values are reset to zero and the BSFC average is set to
    %the value from the previous timestep.
    if (Simulation.BSFC_Energy_Total(timestep) <= 0)
        Simulation.BSFC_Energy_Total(timestep) = 0;
        Simulation.Energy_Pumped(timestep) = 0;
    end

    if (Simulation.Energy_Pumped(timestep) <= 0)
        Simulation.Energy_Pumped(timestep) = 0;
        Simulation.BSFC_AVG(timestep) = Simulation.BSFC_AVG(prevstep);
    else
        Simulation.BSFC_AVG(timestep) = Simulation.BSFC_Energy_Total(timestep) ...
                / Simulation.Energy_Pumped(timestep);
    end


%When there is no net flow into the HP accumulator
else
    Simulation.Energy_Pumped(timestep) = Simulation.Energy_Pumped(prevstep);
    Simulation.BSFC_Energy_Total(timestep) = Simulation.BSFC_Energy_Total(prevstep);
    Simulation.BSFC_AVG(timestep) = Simulation.BSFC_AVG(prevstep);
end


%% Regeneration energy
%Keeps track of energy reclaimed using regen
if (~Simulation.Hydrostatic(prevstep))
    Simulation.Regen_Energy(timestep) = Simulation.Regen_Energy(prevstep) ...
            - Simulation.Motor_Flow_Rate(prevstep)*Simulation.Pressure(prevstep)*Simulation.dt;
    if (Simulation.Regen_Energy(timestep) < 0)
        Simulation.Regen_Energy(timestep) = 0;
    end
else
```

```
    Simulation.Regen_Energy(timestep) = Simulation.Regen_Energy(prevstep);
end



%% Motor
%This section contains the caclculations related to the motor
%
%Section Outline:
%
%Acceleration Positive
%   -Calculate torque required at the differential
%   -Check to see if there is sufficient pressure in the HP ACC to
%       provide the required torque
%   -If the HP ACC can provide the torque:
%       -Calculate minimum effective BSFC using the energy in the HP
%           ACC
%       -Calculate the minimum effective BSFC using the Hydrostatic
%           Mode
%       -Select the more efficient mode
%   -ElseIf the HP ACC can not provide the torque:
%       -Select the Hydrostatic Mode
%
%Acceleration Negative
%   -Calculate torque required at the differential
%   -If the torque required is negative:
%       -Check to see if regen is enabled
%       -If regen is enabled:
%           -Check to see if there is sufficient pressure to provide
%               the required torque
%           -If the required torque can be provided:
%               -Select the gear with the highest flow rate
%           -ElseIf there is insufficient pressure
%               -Select the lowest possible gear and use the mechanical
%                   brakes to provide the rest of the torque
%       -ElseIf regen is disabled:
%           -Use the mechanical brakes to provide the torque
%   -ElseIf the torque required is positive (Wind and rolling
%       resistances can provide more deceleration than desired):
%       -Check to see if there is sufficient pressure in the HP ACC to
```

```
%           provide the required torque
%       -If the HP ACC can provide the torque:
%           -Calculate minimum effective BSFC using the energy in the HP
%               ACC
%           -Calculate the minimum effective BSFC using the Hydrostatic
%               Mode
%           -Select the more efficient mode
%       -ElseIf the HP ACC can not provide the torque:
%           -Select the Hydrostatic Mode
%
%Acceleration Zero and Velocity Positive
%   -Calculate torque required at the differential
%   -Check to see if there is sufficient pressure in the HP ACC to
%       provide the required torque
%   -If the HP ACC can provide the torque:
%       -Calculate minimum effective BSFC using the energy in the HP
%           ACC
%       -Calculate the minimum effective BSFC using the Hydrostatic
%           Mode
%       -Select the more efficient mode
%   -ElseIf the HP ACC can not provide the torque:
%       -Select the Hydrostatic Mode
%
%Acceleration Zero and Velocity Zero
%   -Set operational modes and carry over the gear selection from the
%       previous timestep
% tic
Simulation.Effective_BSFCs = [ ];
if( Simulation.acc > 0)

    %Can't regen on flat ground while accelerating
    Simulation.regen(timestep) = false;

    %Required torque to follow the driving cycle
    %EPA data used for rolling resistance
    Simulation.Torque = ...
            ( ...
            (Vehicle.Weight/32.2 + Vehicle.Wheel_MOI/((Vehicle.Tire_Diameter/2)^2))*Simulation.acc ...
            ...
            + 1/2*Environment.Air_Density*Vehicle.CdA ...
```

```
        *   ( ...
            Simulation.Velocity(timestep)^2 + Simulation.Velocity(timestep) * Simulation.acc ...
            * Simulation.dt+Simulation.acc^2*Simulation.dt^2/3 ...
            ) ...
        ...
        +   ( ...
            Vehicle.RL.C*(Simulation.Velocity(timestep)*3600/5280)^2 ...
            + Vehicle.RL.C*(Simulation.Velocity(timestep)*3600/5280) ...
                * Simulation.acc*Simulation.dt* 3600/5280 ...
            + Vehicle.RL.C*(Simulation.acc*3600/5280)^2*Simulation.dt^2/3 ...
            + Vehicle.RL.B*(Simulation.Velocity(timestep)*3600/5280) ...
            + Vehicle.RL.B*(Simulation.acc*3600/5280)*Simulation.dt/2 + Vehicle.RL.A ...
            ) ...
        ...
        ) ...
        *(Vehicle.Tire_Diameter/2)/Vehicle.Differential_Ratio/Vehicle.Gearbox_Efficiency;


%Check to see if High Pressure accumulator can power the vehicle
Simulation.ACC_Good = false;

Simulation.Motor_Mech_Efficiencies = interp2_mod( ...
        Motor.Speed_Range, ...
        Motor.Pressure_Range, ...
        Motor.Mech_Efficiency_Map, ...
        ( ...
        Simulation.Differential_Speed ...
        + Simulation.acc*Simulation.dt/(2*pi*Vehicle.Tire_Diameter) ...
        * 60*Vehicle.Differential_Ratio ...
        ) ...
        .* Vehicle.Motor_Gear_Ratios, ...
        Simulation.Pressure(timestep)*ones(1,length(Vehicle.Motor_Gear_Ratios)), ...
        'linear',-1);
%The available torque is checked in each motor gear
for index1 = 1:length(Vehicle.Motor_Gear_Ratios)

    Simulation.Speed_Indices = [ ];
    Simulation.Pressure_Indices = [ ];
    Simulation.Speed_Indices = Interp2_Reduction( ...
            Motor.Speed_Range, ...
```

```
            Simulation.Differential_Speed*Vehicle.Motor_Gear_Ratios(index1), ...
            Simulation.Interp2_Range);

    Simulation.Pressure_Indices = Interp2_Reduction( ...
            Motor.Pressure_Range, Simulation.Pressure(timestep), Simulation.Interp2_Range);

    if ( ...
        ( ...
        Simulation.Torque/Vehicle.Motor_Gear_Ratios(index1)/Vehicle.Gearbox_Efficiency ...
        < ...
        (Simulation.Pressure(timestep)*Motor.Displacement_Max/12/2/pi) ...
        *Simulation.Motor_Mech_Efficiencies(index1) ...
        ) ...
        ...
        && ...
        ( ...
        Simulation.Differential_Speed*Vehicle.Motor_Gear_Ratios(index1) ...
        <= ...
        max(Motor.Speed_Range) ...
        ) ...
        ...
        && ...
        ( ...
        Simulation.Differential_Speed*Vehicle.Motor_Gear_Ratios(index1) ...
        >= ...
        Simulation.Regen_Discharge_Min_Motor_Speed) ...
        )


        Simulation.ACC_Good = true;

    end

end

%If the system pressure is sufficient, the Accumulator_Drive code
%is run. The efficiency is then compared to hydrostatic mode to
%determine which mode is selected.
if (Simulation.ACC_Good)
```

```
Accumulator_Drive

%If the Use_Regen_First option is set to true and there is
%energy from regenerative braking available, Hydrostatic mode
%is not used.
if (~(Simulation.Regen_Energy(timestep) && Simulation.Use_Regen_First))

    Simulation.Effective_BSFCs = [ ];
    for index2 = 1:length(Vehicle.EP_Gear_Ratios)
        Simulation.Effective_BSFCs(index2,:) = interp2_mod( ...
                Motor.Speed_Range, ...
                Motor.Torque_Range, ...
                Hydrostatic.Effective_BSFC(:,:,index2), ...
                Simulation.avg_motor_speeds, ...
                Simulation.Torques);
    end

    [Simulation.Effective_BSFCs Simulation.EP_Gears] = min(Simulation.Effective_BSFCs);

    %Check if hybrid mode can supply enough fluid if the HP
    %accumulator is empty and Charge_Enable is true.
    if (Simulation.empty && Simulation.Charge_Enable)

        [Simulation.Effective_BSFC Simulation.EP_Gear_empty] = min( ...
                interp1( ...
                    Pump.Pressure_Range,Hybrid.Effective_BSFC,Simulation.Pressure(timestep) ...
                    ) ...
                );

        Simulation.Pump_Speed_empty = interp1( ...
                Pump.Pressure_Range, ...
                Hybrid.Pump_Speed(:,Simulation.EP_Gear_empty),...
                Simulation.Pressure(timestep));

        Simulation.Pump_Displacement_empty = interp1( ...
                Pump.Pressure_Range, ...
                Hybrid.Pump_Displacement(:,Simulation.EP_Gear_empty), ...
                Simulation.Pressure(timestep));
```

```
    Simulation.Pump_Vol_Efficiency_empty = interp1( ...
            Pump.Pressure_Range,...
            Hybrid.Pump_Vol_Efficiency(:,Simulation.EP_Gear_empty), ...
            Simulation.Pressure(timestep));

    Simulation.Pump_Flow_Rate_empty = Simulation.Pump_Speed_empty ...
            * Simulation.Pump_Displacement_empty*Pump.Displacement_Max ...
            / 60*Simulation.Pump_Vol_Efficiency_empty;

    if ( ...
        ( ...
        min(Simulation.Effective_BSFCs)...
        < ...
        Simulation.BSFC_AVG(timestep) ...
        /   ( ...
            Simulation.Motor_Vol_Efficiency(timestep) ...
            * Simulation.Motor_Mech_Efficiency(timestep)*Vehicle.Gearbox_Efficiency ...
            ) ...
        ) ...
        ...
        && ...
        (Simulation.Motor_Flow_Rate(timestep) > Simulation.Pump_Flow_Rate_empty))


        Hydrostatic_Drive;
    end

%If the HP accumulator is empty and Charge_Enable is false,
%Hydrostatic_Mode is selected
elseif ( Simulation.empty && ~Simulation.Charge_Enable)
        Hydrostatic_Drive;

%If Hydrostatic mode is more efficient than hybrid mode,
%Hydrostatic mode is selected
elseif ( ...
        min(Simulation.Effective_BSFCs) ...
        < ...
        Simulation.BSFC_AVG(timestep)/(Simulation.Motor_Vol_Efficiency(timestep) ...
```

```
                    *Simulation.Motor_Mech_Efficiency(timestep)*Vehicle.Gearbox_Efficiency) ...
                    )

                Hydrostatic_Drive;

            %If hybrid mode is more efficient, the effective BSFC from
            %Hydrostatic mode is stored for post processing
            else
                Simulation.Alt_Effective_BSFC(timestep) = min(Simulation.Effective_BSFCs);
            end

        end

    %There is not sufficient pressure in the system to meet the
    %required torque output. The system is now running in
    %Hydrostatic mode.
    else

        Hydrostatic_Drive

    end




elseif (Simulation.acc < 0)

    %Required torque to follow the driving cycle
    %EPA data used for rolling resistance
    Simulation.Torque = ...
            ( ...
            (Vehicle.Weight/32.2 + Vehicle.Wheel_MOI/((Vehicle.Tire_Diameter/2)^2))*Simulation.acc ...
            ...
            + 1/2*Environment.Air_Density*Vehicle.CdA ...
            *  ( ...
               Simulation.Velocity(timestep)^2+Simulation.Velocity(timestep) ...
               * Simulation.acc*Simulation.dt+Simulation.acc^2*Simulation.dt^2/3 ...
               ) ...
            ...
```

```
        +   ( ...
            Vehicle.RL.C*(Simulation.Velocity(timestep)*3600/5280)^2 ...
            + Vehicle.RL.C*(Simulation.Velocity(timestep)*3600/5280) ...
                * Simulation.acc*Simulation.dt*3600/5280 ...
            + Vehicle.RL.C*(Simulation.acc*3600/5280)^2*Simulation.dt^2/3 ...
            + Vehicle.RL.B*(Simulation.Velocity(timestep)*3600/5280) ...
            + Vehicle.RL.B*(Simulation.acc*3600/5280)*Simulation.dt/2 + Vehicle.RL.A ...
            ) ...
        ) ...
    *(Vehicle.Tire_Diameter/2)/Vehicle.Differential_Ratio;

%The torque is checked for its sign and adjusted by the gearbox
%efficiency
if(Simulation.Torque >= 0)
    Simulation.regen(timestep) = false;
    Simulation.Torque = Simulation.Torque/Vehicle.Gearbox_Efficiency;
elseif ((Simulation.Torque < 0))
    Simulation.regen(timestep) = true;
    Simulation.Hydrostatic(timestep) = false;
    %Simulation.idle(timestep) = true;
    Simulation.Torque = -Simulation.Torque*Vehicle.Gearbox_Efficiency;
end

%if the required torque was negative, regen is set for the timestep
if (Simulation.regen(timestep))

    %Check to see if regenerative braking is enabled
    if (Simulation.Regen_Enable)

        Simulation.Speed_Indicies = [];
        Simulation.Torque_Indicies = [];
        Simulation.Pressure_Indicies = [];
        Simulation.Displacement_Indicies = [];
        Simulation.Pressure_Good = false;

        %Select a range of values around the operating point.
        %Significantly improves interp3 performance
        Simulation.avg_diff_speed = ...
                ( ...
                Simulation.Differential_Speed ...
```

```
        ...
        + Simulation.acc*Simulation.dt ...
        / (2*pi*Vehicle.Tire_Diameter)*60*Vehicle.Differential_Ratio ...
        );
Simulation.avg_motor_speeds = Simulation.avg_diff_speed.*Vehicle.Motor_Gear_Ratios;

Simulation.Pressure_Indicies = Interp3_Reduction( ...
        Motor.Pressure_Range,Simulation.Pressure(timestep),Simulation.Interp3_Range);

%Calculate the motor's mechanical efficiency in each gear
for index1 = 1:length(Vehicle.Motor_Gear_Ratios)
    Simulation.Motor_Mech_Efficiencies(index1) = interp2_mod( ...
        Motor.Speed_Range,Motor.Pressure_Range, ...
        Motor.Mech_Efficiency_Map, ...
        Simulation.avg_motor_speeds(index1),Simulation.Pressure(timestep));
    if (isnan(Simulation.Motor_Mech_Efficiencies(index1)))
       Simulation.Motor_Mech_Efficiencies(index1) = -1;
    end
end

%Calculate the motor torque in each gear
Simulation.Torques = Simulation.Torque./Vehicle.Motor_Gear_Ratios;

for index1 = 1:length(Vehicle.Motor_Gear_Ratios)
    %Check to see if the motor is within its speed range
    if ( ...
        ( ...
        Simulation.Differential_Speed*Vehicle.Motor_Gear_Ratios(index1) ...
        <= ...
        max(Motor.Speed_Range) ...
        ) ...
        && ...
        (Simulation.avg_motor_speeds(index1) <= max(Motor.Speed_Range)) ...
        )

        %Check to see if the motor can provide the required
        %torque
        if ( ...
            Simulation.Torques(index1) ...
            <= ...
```

```
    ( ...
    Simulation.Pressure(timestep)/Simulation.Motor_Mech_Efficiencies(index1)...
    * Motor.Displacement_Max/12/2/pi ...
    ) ...
)

Simulation.Speed_Indicies(index1,:) = Interp3_Reduction( ...
        Motor.Speed_Range, ...
        Simulation.avg_diff_speed*Vehicle.Motor_Gear_Ratios(index1),...
        Simulation.Interp3_Range);

Simulation.Torque_Indicies(index1,:) = Interp3_Reduction( ...
        Motor.Torque_Range,...
        Simulation.Torque/Vehicle.Motor_Gear_Ratios(index1), ...
        Simulation.Interp3_Range);

 %If a full efficiency map is available interp3 is needed to find the
 %volumetric efficieny. This data is not currently available, so a faster
 %piece of code is used instead.

  Simulation.Motor_Displacements(index1) = interp3( ...
      Motor.Speed_Range(Simulation.Speed_Indicies(index1,:)), ...
      Motor.Pressure_Range(Simulation.Pressure_Indicies), ...
      Motor.Torque_Range(Simulation.Torque_Indicies(index1,:)), ...
      Motor.Displacement_Map( ...
          Simulation.Pressure_Indicies, ...
          Simulation.Speed_Indicies(index1,:),...
          Simulation.Torque_Indicies(index1,:) ...
          ), ...
      Simulation.avg_motor_speeds(index1), ...
      Simulation.Pressure(timestep), ...
      Simulation.Torques(index1));

Simulation.Motor_Displacements(index1) = Simulation.Torques(index1)*12*2*pi ...
            / (Simulation.Pressure(timestep)*Motor.Displacement_Max) ...
            * Simulation.Motor_Mech_Efficiencies(index1);
if (Simulation.Motor_Displacements(index1) > 1)
    Simulation.Motor_Displacements(index1) = 0;
end
```

```
%Checks to make sure displacements are valid
if ( ...
    isnan(Simulation.Motor_Displacements(index1)) ...
    && ...
        ( ...
        (Simulation.Torque/Vehicle.Motor_Gear_Ratios(index1)) ...
        <= ...
            ( ...
            (Simulation.Pressure(timestep)*Motor.Displacement_Max/12/2/pi) ...
            / Simulation.Motor_Mech_Efficiencies(index1) ...
            ) ...
        ) ...
    )

    Simulation.Motor_Displacements(index1) = 1;

elseif (isnan(Simulation.Motor_Displacements(index1)))

    Simulation.Motor_Displacements(index1) = Simulation.Torques(index1)*12 ...
            *2*pi ...
            /(Simulation.Pressure(timestep)*Motor.Displacement_Max) ...
            * Simulation.Motor_Mech_Efficiencies(index1);
    if (Simulation.Motor_Displacements(index1) > 1)
        Simulation.Motor_Displacements(index1) = 0;
    end
end



Simulation.Displacement_Indicies(index1,:) = Interp3_Reduction( ...
        Motor.Displacement_Range, ...
        Simulation.Motor_Displacements(index1), ...
        Simulation.Interp3_Range);
Simulation.Motor_Torques(index1) = -Motor.Displacement_Max ...
        * Simulation.Motor_Displacements(index1) ...
        * Simulation.Pressure(timestep) / (12*2*pi) ...
        / Simulation.Motor_Mech_Efficiencies(index1);
```

```
            %If a full efficiency map is available interp3 is needed to find the
            %volumetric efficieny. This data is not currently available, so a faster
            %piece of code is used instead.

%            Simulation.Motor_Vol_Efficiencies(index1) = interp3( ...
%                Motor.Speed_Range(Simulation.Speed_Indicies(index1,:)), ...
%                Motor.Pressure_Range(Simulation.Pressure_Indicies), ...
%                Motor.Displacement_Range(Simulation.Displacement_Indicies(index1,:)), ...
%                Motor.Vol_Efficiency_Map( ...
%                    Simulation.Pressure_Indicies, ...
%                    Simulation.Speed_Indicies(index1,:), ...
%                    Simulation.Displacement_Indicies(index1,:) ...
%                    ), ...
%                Simulation.avg_motor_speeds(index1), ...
%                Simulation.Pressure(timestep), ...
%                Simulation.Motor_Displacements(index1));

            Simulation.Motor_Vol_Efficiencies(index1) = 1 - ( 1- interp2_mod( ...
                    Motor.Speed_Range, ...
                    Motor.Pressure_Range, ...
                    Motor.Vol_Efficiency_Map(:,:,size(Motor.Vol_Efficiency_Map,3)), ...
                    Simulation.avg_motor_speeds(index1), ...
                    Simulation.Pressure(timestep)))/Simulation.Motor_Displacements(index1);

            Simulation.Motor_Flow_Rates(index1) = - Simulation.avg_motor_speeds(index1) ...
                    * Motor.Displacement_Max * Simulation.Motor_Displacements(index1) ...
                    / 60 * Simulation.Motor_Vol_Efficiencies(index1);
            Simulation.Pressure_Good = true;
        else
            %Insufficient pressure to provide required
            %torque. Motor Displacement is set to max.

            %When a lower gear has sufficient pressure to
            %provide the required torque, a taller gear
            %than cannot is not needed.
            if (~Simulation.Pressure_Good)
                Simulation.Motor_Displacements(index1) = 1;
                Simulation.Displacement_Indicies(index1,:) = Interp3_Reduction( ...
                        Motor.Displacement_Range, ...
```

```
            Simulation.Motor_Displacements(index1), ...
            Simulation.Interp3_Range);
    Simulation.Speed_Indicies(index1,:) = Interp3_Reduction( ...
            Motor.Speed_Range, ...
            Simulation.avg_diff_speed*Vehicle.Motor_Gear_Ratios(index1), ...
            Simulation.Interp3_Range);
    Simulation.Motor_Torques(index1) = -Motor.Displacement_Max ...
            * Simulation.Motor_Displacements(index1) ...
            * Simulation.Pressure(timestep)/(12*2*pi) ...
            / Simulation.Motor_Mech_Efficiencies(index1);

%If a full efficiency map is available interp3 is needed to find the
%volumetric efficieny. This data is not currently available, so a faster
%piece of code is used instead.

%     Simulation.Motor_Vol_Efficiencies(index1) = interp3( ...
%             Motor.Speed_Range(Simulation.Speed_Indicies(index1,:)), ...
%             Motor.Pressure_Range(Simulation.Pressure_Indicies), ...
%             Motor.Displacement_Range( ...
%                 Simulation.Displacement_Indicies(index1,:) ...
%                 ), ...
%             Motor.Vol_Efficiency_Map( ...
%                 Simulation.Pressure_Indicies, ...
%                 Simulation.Speed_Indicies(index1,:), ...
%                 Simulation.Displacement_Indicies(index1,:) ...
%                 ), ...
%             Simulation.avg_motor_speeds(index1), ...
%             Simulation.Pressure(timestep), ...
%             Simulation.Motor_Displacements(index1));
    Simulation.Motor_Vol_Efficiencies(index1) = 1 - ( 1- interp2_mod( ...
            Motor.Speed_Range, ...
            Motor.Pressure_Range, ...
            Motor.Vol_Efficiency_Map(:,:,size(Motor.Vol_Efficiency_Map,3)), ...
            Simulation.avg_motor_speeds(index1), ...
            Simulation.Pressure(timestep))) ...
            /Simulation.Motor_Displacements(index1);

    Simulation.Motor_Flow_Rates(index1) = ...
            - Simulation.avg_motor_speeds(index1) ...
            * Motor.Displacement_Max*Simulation.Motor_Displacements(index1) ...
```

```
                        / 60 * Simulation.Motor_Vol_Efficiencies(index1);
                else
                    Simulation.Motor_Displacements(index1) = 0;
                    Simulation.Motor_Torques(index1) = 0;
                    Simulation.Motor_Vol_Efficiencies(index1) = 0;
                    Simulation.Motor_Flow_Rates(index1) = 0;

                end
            end

        else
           %Values are out of range. Null values are set.
           Simulation.Motor_Displacements(index1) = -1;
           Simulation.Motor_Torques(index1) = -1;
           Simulation.Motor_Vol_Efficiencies(index1) = 0;
           Simulation.Motor_Flow_Rates(index1) = inf;

        end

    end


%Mechanical brakes are set when there is insufficient
%pressure to provide the required torque. The lowest
%possible motor gear is selected to maximize regen
%effectiveness
if (~Simulation.Pressure_Good)

    Simulation.Mechanical_Brakes(timestep) = true;
    Simulation.Motor_Displacement(timestep) = 1;
    Simulation.Motor_Gear(timestep) = find(Simulation.Motor_Displacements == 1,1);

    Simulation.Motor_Mech_Efficiency(timestep) = ...
            Simulation.Motor_Mech_Efficiencies(Simulation.Motor_Gear(timestep));

    Simulation.Motor_Torque(timestep) = -Motor.Displacement_Max ...
            * Simulation.Motor_Displacement(timestep)*Simulation.Pressure(timestep) ...
            / (12*2*pi)/Simulation.Motor_Mech_Efficiency(timestep);
```

```
        Simulation.Motor_Vol_Efficiency(timestep) = ...
                Simulation.Motor_Vol_Efficiencies(Simulation.Motor_Gear(timestep));

        Simulation.Motor_Flow_Rate(timestep) = ...
                Simulation.Motor_Flow_Rates(Simulation.Motor_Gear(timestep));

        Simulation.Motor_Speed(timestep) = Simulation.Differential_Speed ...
                * Vehicle.Motor_Gear_Ratios(Simulation.Motor_Gear(timestep));

        Simulation.Motor_Torque(timestep) = ...
                Simulation.Motor_Torques(Simulation.Motor_Gear(timestep));

        if ( ...
            (Simulation.Motor_Vol_Efficiency(timestep) <= 0) ...
            || ...
            (Simulation.Motor_Flow_Rate(timestep) > 0) ...
            )
            %When the motor is leaking more fluid than it is
            %pumping, the motor is disconnected and only the
            %mechanical brakes are used to slow the vehicle.

            Simulation.Motor_Vol_Efficiency(timestep) = 0;
            Simulation.Motor_Torque(timestep) = 0;
            Simulation.Motor_Flow_Rate(timestep) = 0;
            Simulation.Motor_Displacement(timestep) = 0;
            Simulation.Mechanical_Brakes(timestep) = true;

        end

    else
        %When there is sufficient pressure for the motor to
        %provide the required torque, the mechanical brakes are
        %not used.
        Simulation.Mechanical_Brakes(timestep) = false;

        [Simulation.Max_Motor_Flow_Rate Simulation.Motor_Gear(timestep)] = ...
                min(Simulation.Motor_Flow_Rates);
```

```
Simulation.Motor_Displacement(timestep) = ...
        Simulation.Motor_Displacements(Simulation.Motor_Gear(timestep));

if (isnan(Simulation.Motor_Displacement(timestep)))
    %When the displacement is very near 1, the interp3
    %function sometimes returns a NaN due to the
    %discrete nature of the motor tables. The
    %displacement is set to max and the values are
    %recalculated.
    Simulation.Motor_Displacement(timestep) = 1;

    Simulation.Speed_Indicies = [];
    Simulation.Torque_Indicies = [];
    Simulation.Pressure_Indicies = [];
    Simulation.Displacement_Indicies = [];

    Simulation.Speed_Indicies = Interp3_Reduction( ...
            Motor.Speed_Range, ...
            Simulation.avg_motor_speeds(Simulation.Motor_Gear(timestep)),...
            Simulation.Interp3_Range);

    Simulation.Pressure_Indicies = Interp3_Reduction( ...
            Motor.Pressure_Range, ...
            Simulation.Pressure(timestep), ...
            Simulation.Interp3_Range);

    Simulation.Torque_Indicies = Interp3_Reduction( ...
            Motor.Torque_Range, ...
            Simulation.Torques(Simulation.Motor_Gear(timestep)), ...
            Simulation.Interp3_Range);

    Simulation.Displacement_Indicies = Interp3_Reduction( ...
            Motor.Displacement_Range, ...
            Simulation.Motor_Displacement(timestep), ...
            Simulation.Interp3_Range);

    Simulation.Motor_Torque(timestep) = -Motor.Displacement_Max ...
            * Simulation.Motor_Displacement(timestep)*Simulation.Pressure(timestep) ...
            / (12*2*pi)/Simulation.Motor_Mech_Efficiency(timestep);
```

```
        Simulation.Motor_Vol_Efficiency(timestep) = interp3( ...
                Motor.Speed_Range(Simulation.Speed_Indicies(index1,:)), ...
                Motor.Pressure_Range(Simulation.Pressure_Indicies), ...
                Motor.Displacement_Range(Simulation.Displacement_Indicies(index1,:)), ...
                Motor.Vol_Efficiency_Map(Simulation.Pressure_Indicies, ...
                Simulation.Speed_Indicies(index1,:), ...
                Simulation.Displacement_Indicies(index1,:)), ...
                Simulation.avg_motor_speeds(index1), ...
                Simulation.Pressure(timestep), ...
                Simulation.Motor_Displacements(index1));
        Simulation.Motor_Flow_Rate(timestep) = -Simulation.Motor_Speed(timestep) ...
                * Motor.Displacement_Max*Simulation.Motor_Displacement(timestep) ...
                / 60*Simulation.Motor_Vol_Efficiency(timestep);


        Simulation.Motor_Speed(timestep) = Simulation.Differential_Speed ...
                * Vehicle.Motor_Gear_Ratios(Simulation.Motor_Gear(timestep));
    else
        %When the displacement is valid.

        Simulation.Motor_Vol_Efficiency(timestep) = ...
                Simulation.Motor_Vol_Efficiencies(Simulation.Motor_Gear(timestep));
        Simulation.Motor_Mech_Efficiency(timestep) = ...
                Simulation.Motor_Mech_Efficiencies(Simulation.Motor_Gear(timestep));
        Simulation.Motor_Flow_Rate(timestep) = ...
                Simulation.Motor_Flow_Rates(Simulation.Motor_Gear(timestep));
        Simulation.Motor_Torque(timestep) = ...
                Simulation.Motor_Torques(Simulation.Motor_Gear(timestep));
        Simulation.Motor_Speed(timestep) = Simulation.Differential_Speed ...
                * Vehicle.Motor_Gear_Ratios(Simulation.Motor_Gear(timestep));

    end



    if (Simulation.Motor_Vol_Efficiency(timestep) <= 0)
        %When using an equivalent pumping efficiency, the
        %motor efficiency can become negative. If this
```

```
        %happens, the motor is disconnected and the
        %mechanical brakes are applied.
        Simulation.Motor_Vol_Efficiency(timestep) = 0;
        Simulation.Motor_Torque(timestep) = 0;
        Simulation.Motor_Flow_Rate(timestep) = 0;
        Simulation.Motor_Displacement(timestep) = 0;
        Simulation.Mechanical_Brakes(timestep) = true;
    end

end



if (~Simulation.Mechanical_Brakes(timestep))
    %Acceleration is calculated for the timestep
    %EPA road load data is used
    Simulation.Acceleration(timestep) = ...
            ( ...
            Simulation.Motor_Torque(timestep) * Vehicle.Differential_Ratio ...
            * Vehicle.Motor_Gear_Ratios(Simulation.Motor_Gear(timestep)) ...
            / (Vehicle.Gearbox_Efficiency * Vehicle.Tire_Diameter/2) ...
            ...
            - 1/2*Environment.Air_Density*Vehicle.CdA ...
            *   ( ...
                Simulation.Velocity(timestep)^2+Simulation.Velocity(timestep) ...
                * Simulation.acc*Simulation.dt+Simulation.acc^2*Simulation.dt^2/3 ...
                ) ...
            ...
            -   ( ...
                Vehicle.RL.C*(Simulation.Velocity(timestep)*3600/5280)^2 ...
                + Vehicle.RL.C*(Simulation.Velocity(timestep)*3600/5280) ...
                * Simulation.acc*Simulation.dt*3600/5280 ...
                + Vehicle.RL.C*(Simulation.acc*3600/5280)^2*Simulation.dt^2/3 ...
                + Vehicle.RL.B*(Simulation.Velocity(timestep)*3600/5280) ...
                + Vehicle.RL.B*(Simulation.acc*3600/5280)*Simulation.dt/2 ...
                + Vehicle.RL.A ...
                ) ...
            ) ...
            / (Vehicle.Weight/32.2  + Vehicle.Wheel_MOI/((Vehicle.Tire_Diameter/2)^2));
```

```
            %The acceleration is set to the desired acceleration if the
            %mechanical brakes are used
            else
                Simulation.Acceleration(timestep) = Simulation.acc;
            end

        %The mechanical brakes are used when regenerative braking is
        %disabled
        else
            %Regen Disabled.

            Simulation.Mechanical_Brakes(timestep) = true;
            Simulation.regen(timestep) = false;

            Simulation.Motor_Displacement(timestep) = 0;
            Simulation.Motor_Torque(timestep) = 0;
            Simulation.Motor_Vol_Efficiency(timestep) = 0;
            Simulation.Motor_Flow_Rate(timestep) = 0;
            Simulation.Acceleration(timestep) = Simulation.acc;
            Simulation.Motor_Gear(timestep) = Simulation.Motor_Gear(prevstep);

        end

    %The required torque is greater than zero while the acceleration is
    %negative
    else
        %motoring


        %Check to see if High Pressure accumulator can power the
        %vehicle
        Simulation.ACC_Good = false;

        %The available torque is checked in each motor gear
        for index1 = 1:length(Vehicle.Motor_Gear_Ratios)

            Simulation.Speed_Indices = [ ];
            Simulation.Pressure_Indices = [ ];
            Simulation.Speed_Indices = Interp2_Reduction( ...
```

```
        Motor.Speed_Range, ...
        Simulation.Differential_Speed*Vehicle.Motor_Gear_Ratios(index1), ...
        Simulation.Interp2_Range);
Simulation.Pressure_Indices = Interp2_Reduction( ...
        Motor.Pressure_Range, ...
        Simulation.Pressure(timestep), ...
        Simulation.Interp2_Range);

Simulation.Motor_Mech_Efficiencies = interp2_mod( ...
    Motor.Speed_Range, ...
    Motor.Pressure_Range, ...
    Motor.Mech_Efficiency_Map, ...
    ( ...
    Simulation.Differential_Speed ...
    + Simulation.acc*Simulation.dt/(2*pi*Vehicle.Tire_Diameter) ...
    * 60*Vehicle.Differential_Ratio ...
    ) ...
    .* Vehicle.Motor_Gear_Ratios, ...
    Simulation.Pressure(timestep)*ones(1,length(Vehicle.Motor_Gear_Ratios)), ...
    'linear',-1);


if ( ...
    ( ...
    Simulation.Torque/Vehicle.Motor_Gear_Ratios(index1)/Vehicle.Gearbox_Efficiency ...
    < ...
    (Simulation.Pressure(timestep)*Motor.Displacement_Max/12/2/pi) ...
    *Simulation.Motor_Mech_Efficiencies(index1) ...
    ) ...
    ...
    && ...
    (Simulation.Differential_Speed*Vehicle.Motor_Gear_Ratios(index1) ...
    <= ...
    max(Motor.Speed_Range)) ...
    ...
    && ...
    (Simulation.Differential_Speed*Vehicle.Motor_Gear_Ratios(index1) ...
    >= ...
    Simulation.Regen_Discharge_Min_Motor_Speed) ...
   )
```

```
        Simulation.ACC_Good = true;
    end

end


%If the system pressure is sufficient, the Accumulator_Drive code
%is run. The efficiency is then compared to hydrostatic mode to
%determine which mode is selected.
if (Simulation.ACC_Good)
    Accumulator_Drive


    %If the Use_Regen_First option is set to true and there is
    %energy from regenerative braking available, Hydrostatic mode
    %is not used.
    if (~(Simulation.Regen_Energy(timestep) && Simulation.Use_Regen_First))

        Simulation.Effective_BSFCs = [ ];
        for index2 = 1:length(Vehicle.EP_Gear_Ratios)
            Simulation.Effective_BSFCs(index2,:) = interp2_mod( ...
                    Motor.Speed_Range, ...
                    Motor.Torque_Range, ...
                    Hydrostatic.Effective_BSFC(:,:,index2), ...
                    Simulation.avg_motor_speeds, ...
                    Simulation.Torques);
        end
        [Simulation.Effective_BSFCs Simulation.EP_Gears] = min(Simulation.Effective_BSFCs);

        %Check if hybrid mode can supply enough fluid if the HP
        %accumulator is empty and Charge_Enable is true.
        if (Simulation.empty && Simulation.Charge_Enable)

            [Effective_BSFC Simulation.EP_Gear_empty] = min( ...
                    interp1( ...
                        Pump.Pressure_Range, ...
                        Hybrid.Effective_BSFC, ...
```

```
                Simulation.Pressure(timestep) ...
                ) ...
            );
        Simulation.Pump_Speed_empty = interp1( ...
            Pump.Pressure_Range, ...
            Hybrid.Pump_Speed(:,Simulation.EP_Gear_empty), ...
            Simulation.Pressure(timestep));
        Simulation.Pump_Displacement_empty = interp1( ...
            Pump.Pressure_Range,...
            Hybrid.Pump_Displacement(:,Simulation.EP_Gear_empty), ...
            Simulation.Pressure(timestep));
        Simulation.Pump_Vol_Efficiency_empty = interp1( ...
            Pump.Pressure_Range, ...
            Hybrid.Pump_Vol_Efficiency(:,Simulation.EP_Gear_empty), ...
            Simulation.Pressure(timestep));
        Simulation.Pump_Flow_Rate_empty = Simulation.Pump_Speed_empty ...
            * Simulation.Pump_Displacement_empty*Pump.Displacement_Max ...
            / 60*Simulation.Pump_Vol_Efficiency_empty;

        if ( ...
            ( ...
            min(Simulation.Effective_BSFCs) ...
            < ...
            Simulation.BSFC_AVG(timestep) ...
            /   ( ...
                Simulation.Motor_Vol_Efficiency(timestep) ...
                * Simulation.Motor_Mech_Efficiency(timestep)*Vehicle.Gearbox_Efficiency ...
                ) ...
            ) ...
            && ...
            (Simulation.Motor_Flow_Rate(timestep) > Simulation.Pump_Flow_Rate_empty) ...
           )


            Hydrostatic_Drive;
        end

    %If the HP accumulator is empty and Charge_Enable is false,
    %Hydrostatic_Mode is selected
    elseif (Simulation.empty && ~Simulation.Charge_Enable)
```

```
            Hydrostatic_Drive;


        %If Hydrostatic mode is more efficient than hybrid mode,
        %Hydrostatic mode is selected
        elseif ( ...
                min(Simulation.Effective_BSFCs) ...
                < ...
                Simulation.BSFC_AVG(timestep) ...
                /   ( ...
                    Simulation.Motor_Vol_Efficiency(timestep) ...
                    * Simulation.Motor_Mech_Efficiency(timestep)*Vehicle.Gearbox_Efficiency ...
                    ) ...
                )


            Hydrostatic_Drive;

        %If hybrid mode is more efficient, the effective BSFC from
        %Hydrostatic mode is stored for post processing
        else
            Simulation.Alt_Effective_BSFC(timestep) = min(Simulation.Effective_BSFCs);
        end

    end

    %Prevents bad efficiencies when the vehicle is coming to a
    %stop
    if ((Simulation.Velocity(timestep) < 0.01) && (Simulation.acc < 0))
        Simulation.Acceleration(timestep) = Simulation.acc;

        if (Simulation.Motor_Vol_Efficiency(timestep) <= 0)
            %When using an equivalent pumping efficiency, the
            %motor efficiency cab become negative. If this
            %happens, the motor is disconnected and the
            %mechanical brakes are applied.
            Simulation.Motor_Vol_Efficiency(timestep) = 0;
            Simulation.Motor_Torque(timestep) = 0;
            Simulation.Motor_Flow_Rate(timestep) = 0;
```

```
                    Simulation.Motor_Displacement(timestep) = 0;
                    Simulation.Mechanical_Brakes(timestep) = true;
                end

            end


        else

            %There is not sufficient pressure in the system to meet the
            %required torque output. The system is now running in
            %Hydrostatic mode.

            Hydrostatic_Drive

        end

    end


elseif ( (Simulation.Velocity(timestep) > 0) && (Simulation.acc == 0))

    Simulation.regen(timestep) = false;

    %Required torque to follow the driving cycle
    %EPA data used for rolling resistance
    Simulation.Torque = ...
            ( ...
            1/2*Environment.Air_Density*Vehicle.CdA ...
            *   ( ...
                Simulation.Velocity(timestep)^2+Simulation.Velocity(timestep)*Simulation.acc ...
                * Simulation.dt+Simulation.acc^2*Simulation.dt^2/3 ...
                ) ...
            ...
            +   ( ...
                Vehicle.RL.C*(Simulation.Velocity(timestep)*3600/5280)^2 ...
                + Vehicle.RL.C*(Simulation.Velocity(timestep)*3600/5280) ...
                    * Simulation.acc*Simulation.dt*3600/5280 ...
                + Vehicle.RL.C*(Simulation.acc*3600/5280)^2*Simulation.dt^2/3 ...
```

```
            + Vehicle.RL.B*(Simulation.Velocity(timestep)*3600/5280) ...
            + Vehicle.RL.B*(Simulation.acc*3600/5280)*Simulation.dt/2 + Vehicle.RL.A ...
            ) ...
        ) ...
    *(Vehicle.Tire_Diameter/2)/Vehicle.Differential_Ratio/Vehicle.Gearbox_Efficiency;

Simulation.ACC_Good = false;

Simulation.Motor_Mech_Efficiencies = interp2_mod( ...
            Motor.Speed_Range, ...
            Motor.Pressure_Range, ...
            Motor.Mech_Efficiency_Map, ...
            ( ...
            Simulation.Differential_Speed ...
            + Simulation.acc*Simulation.dt/(2*pi*Vehicle.Tire_Diameter) ...
            * 60*Vehicle.Differential_Ratio ...
            ) ...
            .* Vehicle.Motor_Gear_Ratios, ...
            Simulation.Pressure(timestep)*ones(1,length(Vehicle.Motor_Gear_Ratios)), ...
            'linear',-1);


%The available torque is checked in each motor gear
for index1 = 1:length(Vehicle.Motor_Gear_Ratios)

    Simulation.Speed_Indices = [ ];
    Simulation.Pressure_Indices = [ ];

    Simulation.Speed_Indices = Interp2_Reduction( ...
            Motor.Speed_Range, ...
            Simulation.Differential_Speed*Vehicle.Motor_Gear_Ratios(index1), ...
            Simulation.Interp2_Range);

    Simulation.Pressure_Indices = Interp2_Reduction( ...
            Motor.Pressure_Range, ...
            Simulation.Pressure(timestep), ...
            Simulation.Interp2_Range);

    if ( ...
```

```
            ( ...
            Simulation.Torque/Vehicle.Motor_Gear_Ratios(index1)/Vehicle.Gearbox_Efficiency ...
            < ...
            (Simulation.Pressure(timestep)*Motor.Displacement_Max/12/2/pi) ...
            * Simulation.Motor_Mech_Efficiencies(index1) ...
            ) ...
            && ...
            ( ...
            Simulation.Differential_Speed*Vehicle.Motor_Gear_Ratios(index1) ...
            <= ...
            max(Motor.Speed_Range) ...
            ) ...
            && ...
            ( ...
            Simulation.Differential_Speed*Vehicle.Motor_Gear_Ratios(index1) ...
            >= ...
            Simulation.Regen_Discharge_Min_Motor_Speed ...
            ) ...
           )

            Simulation.ACC_Good = true;
        end


    end


    %If the system pressure is sufficient, the Accumulator_Drive code
    %is run. The efficiency is then compared to hydrostatic mode to
    %determine which mode is selected.
    if (Simulation.ACC_Good)

        Accumulator_Drive


        if (~(Simulation.Regen_Energy(timestep) && Simulation.Use_Regen_First))

            Simulation.Effective_BSFCs = [ ];
            for index2 = 1:length(Vehicle.EP_Gear_Ratios)
```

```
    Simulation.Effective_BSFCs(index2,:) = interp2_mod( ...
            Motor.Speed_Range, ...
            Motor.Torque_Range, ...
            Hydrostatic.Effective_BSFC(:,:,index2), ...
            Simulation.avg_motor_speeds, ...
            Simulation.Torques);
end

[Simulation.Effective_BSFCs Simulation.EP_Gears] = min(Simulation.Effective_BSFCs);


%Check if hybrid mode can supply enough fluid if the HP
%accumulator is empty and Charge_Enable is true.
if (Simulation.empty && Simulation.Charge_Enable)

    [Effective_BSFC Simulation.EP_Gear_empty] = min( ...
            interp1( ...
                Pump.Pressure_Range, ...
                Hybrid.Effective_BSFC, ...
                Simulation.Pressure(timestep) ...
                ) ...
            );

    Simulation.Pump_Speed_empty = interp1( ...
            Pump.Pressure_Range, ...
            Hybrid.Pump_Speed(:,Simulation.EP_Gear_empty), ...
            Simulation.Pressure(timestep));

    Simulation.Pump_Displacement_empty = interp1( ...
            Pump.Pressure_Range, ...
            Hybrid.Pump_Displacement(:,Simulation.EP_Gear_empty), ...
            Simulation.Pressure(timestep));

    Simulation.Pump_Vol_Efficiency_empty = interp1( ...
            Pump.Pressure_Range, ...
            Hybrid.Pump_Vol_Efficiency(:,Simulation.EP_Gear_empty), ...
            Simulation.Pressure(timestep));

    Simulation.Pump_Flow_Rate_empty = Simulation.Pump_Speed_empty ...
```

```
            * Simulation.Pump_Displacement_empty*Pump.Displacement_Max ...
            / 60*Simulation.Pump_Vol_Efficiency_empty;

    if ( ...
        ( ...
        min(Simulation.Effective_BSFCs) ...
        < ...
        Simulation.BSFC_AVG(timestep) ...
        /   ( ...
            Simulation.Motor_Vol_Efficiency(timestep) ...
            * Simulation.Motor_Mech_Efficiency(timestep) * Vehicle.Gearbox_Efficiency ...
            ) ...
        ) ...
        && ...
        (Simulation.Motor_Flow_Rate(timestep) > Simulation.Pump_Flow_Rate_empty) ...
        )


        Hydrostatic_Drive;
    end


%If the HP accumulator is empty and Charge_Enable is false,
%Hydrostatic_Mode is selected
elseif (Simulation.empty && ~Simulation.Charge_Enable)
    Hydrostatic_Drive;

%If Hydrostatic mode is more efficient than hybrid mode,
%Hydrostatic mode is selected
elseif ( ...
        min(Simulation.Effective_BSFCs) ...
        < ...
        Simulation.BSFC_AVG(timestep) ...
        /   ( ...
            Simulation.Motor_Vol_Efficiency(timestep) ...
            * Simulation.Motor_Mech_Efficiency(timestep)*Vehicle.Gearbox_Efficiency ...
            ) ...
        )
```

```
                Hydrostatic_Drive;

            %If hybrid mode is more efficient, the effective BSFC from
            %Hydrostatic mode is stored for post processing
            else
                Simulation.Alt_Effective_BSFC(timestep) = min(Simulation.Effective_BSFCs);
            end

        end

    %There is not sufficient pressure in the system to meet the
    %required torque output. The system is now running in
    %Hydrostatic mode.
    else

        Hydrostatic_Drive

    end

%The vehicle is stopped and does not need to accelerate
else
    Simulation.idle(timestep) = true;
    Simulation.Hydrostatic(timestep) = false;
    Simulation.regen(timestep) = false;
    Simulation.Acceleration(timestep) = 0;
    Simulation.Motor_Gear(timestep) = Simulation.Motor_Gear(prevstep);



end

Simulation.Position(timestep) = Simulation.Position(prevstep) ...
        + Simulation.Velocity(prevstep)*Simulation.dt;


%% Engine/Pump System
%This section contains the calculations for the combined engine/pump
```

```
%system
%
%Section Outline:
%
%If the pressure is lower than the charge pressure and the system is
%   set to charge or the accumulator is empty and the pressure is not
%   low and the system is not in hydrostatic mode and the system
%   hasn't exceeded the maximum pressure and charging the accumulator
%   is enabled:
%   -Set operation mode
%   -Calculate most efficient operating point
%ElseIf the pressure is low and the system is not in hydrostatic mode
%   -Set operation mode
%   -Calculate highest flow rate possible
%ElseIf Hydrostatic mode
%   -Calculate Engine/Pump parameters based on motor operation and gear
%       selection
%Else idle
%   -Set idle conditions
%   -Check system pressure
%   -If low pressure:
%       -Set operation mode to charge
if ( ...
    (Simulation.Pressure(timestep) < Simulation.Charge_Pressure) ...
    && ...
    (Simulation.charge(prevstep) || Simulation.empty) ...
    && ...
    ~Simulation.Pressure_Low ...
    && ...
    ~Simulation.Hydrostatic(timestep) ...
    && ...
    Simulation.Charge_Enable ...
    && ...
    ~Simulation.Over_Pressure ...
    )

    Simulation.idle(timestep) = false;
    if (Simulation.Pressure(timestep) < Simulation.Charge_Pressure)
        Simulation.charge(timestep) = true;
    end
```

```
[Effective_BSFC Simulation.EP_Gear(timestep)] = min( ...
        interp1(Pump.Pressure_Range, ...
            Hybrid.Effective_BSFC, ...
            Simulation.Pressure(timestep) ...
            ) ...
        );

Simulation.Engine_Speed(timestep) = interp1( ...
        Pump.Pressure_Range, ...
        Hybrid.Engine_Speed(:,Simulation.EP_Gear(timestep)), ...
        Simulation.Pressure(timestep));

Simulation.Engine_Torque(timestep) = interp1( ...
        Pump.Pressure_Range, ...
        Hybrid.Engine_Torque(:,Simulation.EP_Gear(timestep)), ...
        Simulation.Pressure(timestep));

Simulation.Engine_BSFC(timestep) = interp1( ...
        Pump.Pressure_Range, ...
        Hybrid.Engine_BSFC(:,Simulation.EP_Gear(timestep)), ...
        Simulation.Pressure(timestep));

Simulation.Pump_Speed(timestep) = interp1( ...
        Pump.Pressure_Range, ...
        Hybrid.Pump_Speed(:,Simulation.EP_Gear(timestep)), ...
        Simulation.Pressure(timestep));

Simulation.Pump_Displacement(timestep) = interp1( ...
        Pump.Pressure_Range, ...
        Hybrid.Pump_Displacement(:,Simulation.EP_Gear(timestep)), ...
        Simulation.Pressure(timestep));

Simulation.Pump_Vol_Efficiency(timestep) = interp1( ...
        Pump.Pressure_Range, ...
        Hybrid.Pump_Vol_Efficiency(:,Simulation.EP_Gear(timestep)), ...
        Simulation.Pressure(timestep));
```

```
    Simulation.Pump_Mech_Efficiency(timestep) = interp1( ...
            Pump.Pressure_Range, ...
            Hybrid.Pump_Mech_Efficiency(:,Simulation.EP_Gear(timestep)), ...
            Simulation.Pressure(timestep));


    Simulation.Pump_Flow_Rate(timestep) = Simulation.Pump_Speed(timestep) ...
            * Simulation.Pump_Displacement(timestep)*Pump.Displacement_Max ...
            / 60*Simulation.Pump_Vol_Efficiency(timestep);

    Simulation.Fuel_Mass(timestep) = Simulation.Engine_BSFC(timestep) ...
            * Simulation.Engine_Speed(timestep) * Simulation.Engine_Torque(timestep) ...
            * (2*pi/60/550)*Simulation.dt/3600;

    Simulation.Fuel_Mass_Total(timestep) = Simulation.Fuel_Mass_Total(prevstep) ...
            + Simulation.Fuel_Mass(timestep);


%Low pressure and not hydrostatic mode
elseif (Simulation.Pressure_Low && ~Simulation.Hydrostatic(timestep))
    Simulation.idle(timestep) = false;

    Simulation.Pump_Flow_Rate_Interp(1:length(EP_Torque_Range),1:length(Pump_Speed_Range))=0;

    for index2 = 1:length(Engine.Torque_Range)

        Simulation.Pump_Flow_Rate_Interp(index2,:) = interp1( ...
                Pump.Pressure_Range, ...
                Pump.Flow_Rate_Map(:,:,index2), ...
                Simulation.Pressure(timestep));
    end

    for index1 = 1:size(Simulation.Pump_Flow_Rate_Interp,1)
        for index2 = 1:size(Simulation.Pump_Flow_Rate_Interp,2)
            if ( isnan(Simulation.Pump_Flow_Rate_Interp(index1,index2)))
                Simulation.Pump_Flow_Rate_Interp(index1,index2)=0;
            end
        end
```

```
end

[Simulation.Column_Maxs Simulation.Row_Index] = max(Simulation.Pump_Flow_Rate_Interp);
[Simulation.Maximum Simulation.Column_Index] = max(Simulation.Column_Maxs);


Simulation.Engine_Speed(timestep) = Engine.Speed_Range(Simulation.Column_Index);
Simulation.Engine_Torque(timestep) = ...
        Engine.Torque_Range(Simulation.Row_Index(Simulation.Column_Index));
Simulation.Engine_BSFC(timestep) = interp2_mod( ...
        Engine.Speed_Range, ...
        Engine.Torque_Range, ...
        Engine.BSFC_Map, ...
        Simulation.Engine_Speed(timestep), ...
        Simulation.Engine_Torque(timestep));

Simulation.Pump_Flow_Rate_Interp = [];

Simulation.Pump_Speed(timestep) = Simulation.Engine_Speed(timestep);
Simulation.Pump_Displacement(timestep) = interp3( ...
        Pump.Speed_Range, ...
        Pump.Pressure_Range, ...
        Engine.Torque_Range, ...
        Pump.Displacement_Map, ...
        Simulation.Pump_Speed(timestep), ...
        Simulation.Pressure(timestep), ...
        Simulation.Engine_Torque(timestep));


Simulation.Pump_Vol_Efficiency(timestep) = interp3( ...
        Pump.Speed_Range, ...
        Pump.Pressure_Range, ...
        Pump.Displacement_Range, ...
        Pump.Vol_Efficiency_Map, ...
        Simulation.Pump_Speed(timestep), ...
        Simulation.Pressure(timestep), ...
        Simulation.Pump_Displacement(timestep));

Simulation.Pump_Mech_Efficiency(timestep) = interp2_mod( ...
```

```
            Pump.Speed_Range, ...
            Pump.Pressure_Range, ...
            Pump.Mech_Efficiency_Map, ...
            Simulation.Pump_Speed(timestep), ...
            Simulation.Pressure(timestep));

    Simulation.Pump_Flow_Rate(timestep) = Simulation.Pump_Speed(timestep) ...
            * Simulation.Pump_Displacement(timestep)*Pump.Displacement_Max ...
            / 60*Simulation.Pump_Vol_Efficiency(timestep);


    Simulation.Fuel_Mass(timestep) = Simulation.Engine_BSFC(timestep) ...
            * Simulation.Engine_Speed(timestep)*Simulation.Engine_Torque(timestep) ...
            * (2*pi/60/550)*Simulation.dt/3600;

    Simulation.Fuel_Mass_Total(timestep) = Simulation.Fuel_Mass_Total(prevstep) ...
            + Simulation.Fuel_Mass(timestep);

%Hydrostatic mode
elseif (Simulation.Hydrostatic(timestep))
    Simulation.Engine_Speed(timestep) = interp2_mod( ...
            Motor.Speed_Range, ...
            Motor.Torque_Range, ...
            Hydrostatic.Engine_Speed(:,:,Simulation.EP_Gear(timestep)), ...
            Simulation.avg_motor_speed, ...
            Simulation.Torque);

    Simulation.Engine_Torque(timestep) = interp2_mod( ...
            Motor.Speed_Range, ...
            Motor.Torque_Range, ...
            Hydrostatic.Engine_Torque(:,:,Simulation.EP_Gear(timestep)), ...
            Simulation.avg_motor_speed,Simulation.Torque);

    Simulation.Engine_BSFC(timestep) = interp2_mod( ...
            Motor.Speed_Range, ...
            Motor.Torque_Range, ...
            Hydrostatic.Engine_BSFC(:,:,Simulation.EP_Gear(timestep)), ...
            Simulation.avg_motor_speed, ...
            Simulation.Torque);
```

```
    Simulation.Pump_Speed(timestep) = Simulation.Engine_Speed(timestep) ...
            /Hydrostatic.EP_Gear_Ratios(Simulation.EP_Gear(timestep));
    Simulation.Pump_Displacement(timestep) = interp2_mod( ...
            Motor.Speed_Range, ...
            Motor.Torque_Range, ...
            Hydrostatic.Pump_Disp(:,:,Simulation.EP_Gear(timestep)), ...
            Simulation.avg_motor_speed, ...
            Simulation.Torque);

    Simulation.Pump_Vol_Efficiency(timestep) = interp2_mod( ...
        Motor.Speed_Range, ...
        Motor.Torque_Range, ...
        Hydrostatic.Pump_Vol_Efficiency(:,:,Simulation.EP_Gear(timestep)), ...
        Simulation.avg_motor_speed, ...
        Simulation.Torque);

    Simulation.Pump_Mech_Efficiency(timestep) = interp2_mod( ...
            Motor.Speed_Range, ...
            Motor.Torque_Range, ...
            Hydrostatic.Pump_Mech_Efficiency(:,:,Simulation.EP_Gear(timestep)), ...
            Simulation.avg_motor_speed, ...
            Simulation.Torque);

    Simulation.Pump_Flow_Rate(timestep) = Simulation.Motor_Flow_Rate(timestep);

    Simulation.Fuel_Mass(timestep) = Simulation.Engine_BSFC(timestep) ...
            * Simulation.Engine_Speed(timestep)*Simulation.Engine_Torque(timestep) ...
            * (2*pi/60/550)*Simulation.dt/3600;

    Simulation.Fuel_Mass_Total(timestep) = Simulation.Fuel_Mass_Total(prevstep) ...
            + Simulation.Fuel_Mass(timestep);

%The engine is idling
else
    %Simulation.idle = true;
    Simulation.Engine_Speed(timestep) = 1000;
    Simulation.Engine_Torque(timestep) = 10;
    Simulation.Engine_BSFC(timestep) = 0;
```

```
    Simulation.EP_Gear(timestep) = Simulation.EP_Gear(prevstep);


    Simulation.Pump_Speed(timestep) = Simulation.Engine_Speed(timestep) ...
        / Hydrostatic.EP_Gear_Ratios(Simulation.EP_Gear(timestep));
    Simulation.Pump_Displacement(timestep) = 0;
    Simulation.Pump_Vol_Efficiency(timestep) = 0;
    Simulation.Pump_Mech_Efficiency(timestep) = 0;

    Simulation.Pump_Flow_Rate(timestep) = 0;

    if(Simulation.Pressure(timestep) < (Hydraulics.Precharge + Simulation.Pressure_Buffer))
        Simulation.charge(timestep) = true;
        Simulation.idle(timestep) = true;
    elseif( ...
            Simulation.Pressure(timestep) ...
            > ...
            Simulation.Charge_Pressure ...
            || ...
            (Simulation.charge(timestep) == false) ...
            )

        Simulation.charge(timestep) = false;
        Simulation.idle(timestep) = true;
    end

    if (Simulation.Engine_Shutdown)
        Simulation.Fuel_Mass(timestep) = 0;
    else
        Simulation.Fuel_Mass(timestep) = Engine.FFR_Idle*2.2/3600*Simulation.dt;
    end

    Simulation.Fuel_Mass_Total(timestep) = Simulation.Fuel_Mass_Total(prevstep) ...
            + Simulation.Fuel_Mass(timestep);

    Simulation.idle_fuel = Simulation.idle_fuel + Simulation.Fuel_Mass(timestep);
end
```

```
%calculate fuel mileage
Simulation.MPG_Current(timestep) = (Simulation.Position(timestep)-Simulation.Position(prevstep)) ...
        / 5280/Simulation.Fuel_Mass(timestep)*Environment.Diesel_Density;
Simulation.MPG_Average(timestep) = Simulation.Position(timestep)/5280 ...
        / Simulation.Fuel_Mass_Total(timestep)*Environment.Diesel_Density;



end
% Simulation Ends

%cleanup fields
if (isfield(Simulation, 'Temperature_Indices'))
    Simulation = rmfield(Simulation, 'Temperature_Indices');
end
if (isfield(Simulation, 'Differential_Speed'))
    Simulation = rmfield(Simulation, 'Differential_Speed');
end
if (isfield(Simulation, 'acc'))
    Simulation = rmfield(Simulation, 'acc');
end
if (isfield(Simulation, 'Torque'))
    Simulation = rmfield(Simulation, 'Torque');
end
if (isfield(Simulation, 'ACC_Good'))
    Simulation = rmfield(Simulation, 'ACC_Good');
end
if (isfield(Simulation, 'Speed_Indices'))
    Simulation = rmfield(Simulation, 'Speed_Indices');
end
if (isfield(Simulation, 'Pressure_Indices'))
    Simulation = rmfield(Simulation, 'Pressure_Indices');
end
if (isfield(Simulation, 'avg_diff_speed'))
    Simulation = rmfield(Simulation, 'avg_diff_speed');
end
if (isfield(Simulation, 'avg_motor_speeds'))
    Simulation = rmfield(Simulation, 'avg_motor_speeds');
end
```

```
if (isfield(Simulation, 'Torques'))
    Simulation = rmfield(Simulation, 'Torques');
end
if (isfield(Simulation, 'Speed_Indicies'))
    Simulation = rmfield(Simulation, 'Speed_Indicies');
end
if (isfield(Simulation, 'Torque_Indicies'))
    Simulation = rmfield(Simulation, 'Torque_Indicies');
end
if (isfield(Simulation, 'Pressure_Indicies'))
    Simulation = rmfield(Simulation, 'Pressure_Indicies');
end
if (isfield(Simulation, 'Displacement_Indicies'))
    Simulation = rmfield(Simulation, 'Displacement_Indicies');
end
if (isfield(Simulation, 'Motor_Mech_Efficiencies'))
    Simulation = rmfield(Simulation, 'Motor_Mech_Efficiencies');
end
if (isfield(Simulation, 'Motor_Displacements'))
    Simulation = rmfield(Simulation, 'Motor_Displacements');
end
if (isfield(Simulation, 'Motor_Torques'))
    Simulation = rmfield(Simulation, 'Motor_Torques');
end
if (isfield(Simulation, 'Accelerations'))
    Simulation = rmfield(Simulation, 'Accelerations');
end
if (isfield(Simulation, 'Motor_Vol_Efficiencies'))
    Simulation = rmfield(Simulation, 'Motor_Vol_Efficiencies');
end
if (isfield(Simulation, 'Motor_Flow_Rates'))
    Simulation = rmfield(Simulation, 'Motor_Flow_Rates');
end
if (isfield(Simulation, 'Max_Motor_Efficiency'))
    Simulation = rmfield(Simulation, 'Max_Motor_Efficiency');
end
if (isfield(Simulation, 'Torque_Indicies'))
    Simulation = rmfield(Simulation, 'Torque_Indicies');
end
if (isfield(Simulation, 'Effective_BSFCs'))
    Simulation = rmfield(Simulation, 'Effective_BSFCs');
```

```
    end
    if (isfield(Simulation, 'avg_motor_speed'))
        Simulation = rmfield(Simulation, 'avg_motor_speed');
    end
    if (isfield(Simulation, 'Max_Motor_Flow_Rate'))
        Simulation = rmfield(Simulation, 'Max_Motor_Flow_Rate');
    end
    if (isfield(Simulation, 'Motor_Mech_Efficiencies'))
        Simulation = rmfield(Simulation, 'Motor_Mech_Efficiencies');
    end
    if (isfield(Simulation, 'Pressure_Good'))
        Simulation = rmfield(Simulation, 'Pressure_Good');
    end
    if (isfield(Simulation, 'empty'))
        Simulation = rmfield(Simulation, 'empty');
    end
    if (isfield(Simulation, 'Pressure_Low'))
        Simulation = rmfield(Simulation, 'Pressure_Low');
    end


    simout = Simulation;
end
```

Appendix S: Accumulator_Drive.m

```
%% Accumulator_Drive.m
% This file determines the motor parameters when driving in hybrid mode.
% Commented code is used if full efficiency maps are available.


%Set mode values
Simulation.Hydrostatic(timestep) = false;
Simulation.Pressure_Low = false;


%Select a range of values around the operating point.
%Significantly improves interp3 performance
Simulation.avg_diff_speed = ( ...
        Simulation.Differential_Speed ...
        + Simulation.acc*Simulation.dt ...
        /(2*pi*Vehicle.Tire_Diameter)*60*Vehicle.Differential_Ratio ...
        );
Simulation.avg_motor_speeds = ...
        Simulation.avg_diff_speed.*Vehicle.Motor_Gear_Ratios;

%Gearbox efficiency has already been acounted for
Simulation.Torques = Simulation.Torque./Vehicle.Motor_Gear_Ratios;

Simulation.Speed_Indicies = [];
Simulation.Torque_Indicies = [];
Simulation.Pressure_Indicies = [];
Simulation.Displacement_Indicies = [];

Simulation.Pressure_Indicies = Interp_reduction( ...
        Motor.Pressure_Range, ...
        Simulation.Pressure(timestep), ...
        Simulation.Interp3_Range);


for index1 = 1:length(Vehicle.Motor_Gear_Ratios)
    Simulation.Speed_Indicies(index1,:) = Interp_reduction( ...
            Motor.Speed_Range, ...
            Simulation.avg_diff_speed*Vehicle.Motor_Gear_Ratios(index1), ...
            Simulation.Interp3_Range);

    Simulation.Torque_Indicies(index1,:) = Interp_reduction( ...
            Motor.Torque_Range, ...
            Simulation.Torque/Vehicle.Motor_Gear_Ratios(index1), ...
            Simulation.Interp3_Range);

%     Simulation.Motor_Mech_Efficiencies(index1) = interp2_mod( ...
%             Motor.Speed_Range, ...
%             Motor.Pressure_Range,...
%             Motor.Mech_Efficiency_Map, ...
%             Simulation.avg_motor_speeds(index1), ...
%             Simulation.Pressure(timestep));
%     Simulation.Motor_Mech_Efficiencies(index1) = interp2_mod( ...
%             Motor.Speed_Range(Simulation.Speed_Indicies(index1,:)), ...
```

```
%             Motor.Pressure_Range(Simulation.Pressure_Indicies), ...
%             Motor.Mech_Efficiency_Map( ...
%                 Simulation.Pressure_Indicies, ...
%                 Simulation.Speed_Indicies(index1,:)), ...
%             Simulation.avg_motor_speeds(index1), ...
%             Simulation.Pressure(timestep));
%     Simulation.Motor_Displacements(index1) = interp3( ...
%             Motor.Speed_Range(Simulation.Speed_Indicies(index1,:)), ...
%             Motor.Pressure_Range(Simulation.Pressure_Indicies), ...
%             Motor.Torque_Range(Simulation.Torque_Indicies(index1,:)), ...
%             Motor.Displacement_Map( ...
%                 Simulation.Pressure_Indicies, ...
%                 Simulation.Speed_Indicies(index1,:), ...
%                 Simulation.Torque_Indicies(index1,:)), ...
%             Simulation.avg_motor_speeds(index1), ...
%             Simulation.Pressure(timestep), ...
%             Simulation.Torques(index1));
```

```
    Simulation.Motor_Displacements(index1) = ...
            Simulation.Torques(index1)*12*2*pi ...
            /   ( ...
                Simulation.Motor_Mech_Efficiencies(index1) ...
                * Simulation.Pressure(timestep)*Motor.Displacement_Max ...
                );
    if (Simulation.Motor_Displacements(index1) > 1)
        Simulation.Motor_Displacements(index1) = 0;
    end
    %interp3 can return NaN if the displacement is close to 1
    if ( ...
            isnan(Simulation.Motor_Displacements(index1)) ...
            && ...
            ( ...
                (Simulation.Torque/Vehicle.Motor_Gear_Ratios(index1)) ...
                <= ...
                ( ...
                    ( ...
                        Simulation.Pressure(timestep)* ...
                        Motor.Displacement_Max/12/2/pi ...
                    ) * Simulation.Motor_Mech_Efficiencies(index1) ...
                ) ...
            ) ...
        )
        %Simulation.Motor_Displacements(index1) = 1;

        Simulation.Motor_Displacements(index1) = ...
                Simulation.Torques(index1)*12*2*pi ...
                /   ( ...
                    Simulation.Motor_Mech_Efficiencies(index1) ...
                    * Simulation.Pressure(timestep)*Motor.Displacement_Max...
                    );
        if (Simulation.Motor_Displacements(index1) > 1)
            Simulation.Motor_Displacements(index1) = 0;
```

```
    end

elseif (isnan(Simulation.Motor_Displacements(index1)))
    Simulation.Motor_Displacements(index1) = 0;
end


Simulation.Displacement_Indicies(index1,:) = Interp_reduction( ...
        Motor.Displacement_Range, ...
        Simulation.Motor_Displacements(index1), ...
        Simulation.Interp3_Range);

Simulation.Motor_Torques(index1) = Motor.Displacement_Max ...
        * Simulation.Motor_Displacements(index1) ...
        * Simulation.Pressure(timestep)/(12*2*pi) ...
        * Simulation.Motor_Mech_Efficiencies(index1);

Simulation.Accelerations(index1) = ...
        ( ...
            Simulation.Motor_Torques(index1) ...
            *Vehicle.Differential_Ratio*Vehicle.Gearbox_Efficiency ...
            *Vehicle.Motor_Gear_Ratios(index1) ...
            / (Vehicle.Tire_Diameter/2) ...
            -1/2*Environment.Air_Density*Vehicle.CdA ...
                *( ...
                    Simulation.Velocity(timestep)^2 ...
                    + Simulation.Velocity(timestep)*Simulation.acc ...
                        * Simulation.dt ...
                    + Simulation.acc^2*Simulation.dt^2/3 ...
                ) ...
            - ( ...
                Vehicle.RL.C ...
                    *(Simulation.Velocity(timestep)*3600/5280)^2 ...
                + Vehicle.RL.C ...
                    * (Simulation.Velocity(timestep)*3600/5280) ...
                    * Simulation.acc*Simulation.dt*3600/5280 ...
                + Vehicle.RL.C ...
                    * (Simulation.acc*3600/5280)^2*Simulation.dt^2/3 ...
                + Vehicle.RL.B ...
                    * (Simulation.Velocity(timestep)*3600/5280) ...
                + Vehicle.RL.B ...
                    * (Simulation.acc*3600/5280)*Simulation.dt/2 ...
                + Vehicle.RL.A ...
              ) ...
        ) ...
        / ( ...
            Vehicle.Weight/32.2  ...
            + Vehicle.Wheel_MOI/((Vehicle.Tire_Diameter/2)^2) ...
          );

%     Simulation.Motor_Vol_Efficiencies(index1) = interp3( ...
%             Motor.Speed_Range(Simulation.Speed_Indicies(index1,:)), ...
%             Motor.Pressure_Range(Simulation.Pressure_Indicies), ...
%             Motor.Displacement_Range( ...
%                 Simulation.Displacement_Indicies(index1,:)), ...
%             Motor.Vol_Efficiency_Map( ...
```

```
%                 Simulation.Pressure_Indicies, ...
%                 Simulation.Speed_Indicies(index1,:), ...
%                 Simulation.Displacement_Indicies(index1,:)), ...
%             Simulation.avg_motor_speeds(index1), ...
%             Simulation.Pressure(timestep), ...
%             Simulation.Motor_Displacements(index1));
%     Simulation.Motor_Flow_Rates(index1) = ...
%             Simulation.avg_motor_speeds(index1)*Motor.Displacement_Max ...
%             * Simulation.Motor_Displacements(index1)/60 ...
%             / Simulation.Motor_Vol_Efficiencies(index1);
end

%Using constant leakage assumption
Simulation.Motor_Vol_Efficiencies = ...
        1 ...
        ./ ( ...
            1 ...
            + ( ...
                1 ...
                ./ interp2_mod( ...
                    Motor.Speed_Range,...
                    Motor.Pressure_Range, ...
                    Motor.Vol_Efficiency_Map( ...
                        :,:,size(Motor.Vol_Efficiency_Map,3)), ...
                    Simulation.avg_motor_speeds, ...
                    Simulation.Pressure(timestep) ...
                        *ones(1,length(Vehicle.Motor_Gear_Ratios))...
                    ) ...
                    - 1 ...
              ) ...
            ./ Simulation.Motor_Displacements ...
        );

Simulation.Motor_Flow_Rates = Simulation.avg_motor_speeds ...
        .*Motor.Displacement_Max.*Simulation.Motor_Displacements ...
        ./60./Simulation.Motor_Vol_Efficiencies;

%If there is at least one valid displacement
if (sum(Simulation.Motor_Displacements) > 0)
    [Simulation.Max_Motor_Efficiency Simulation.Motor_Gear(timestep)] = ...
        max( ...
            Simulation.Motor_Mech_Efficiencies ...
            .*Simulation.Motor_Vol_Efficiencies);

    Simulation.Motor_Displacement(timestep) = ...
        Simulation.Motor_Displacements(Simulation.Motor_Gear(timestep));

    Simulation.Motor_Mech_Efficiency(timestep) = ...
        Simulation.Motor_Mech_Efficiencies(Simulation.Motor_Gear(timestep));

    Simulation.Motor_Torque(timestep) = ...
        Simulation.Motor_Torques(Simulation.Motor_Gear(timestep));

    Simulation.Motor_Vol_Efficiency(timestep) = ...
        Simulation.Motor_Vol_Efficiencies(Simulation.Motor_Gear(timestep));
```

```
    Simulation.Motor_Flow_Rate(timestep) = ...
        Simulation.Motor_Flow_Rates(Simulation.Motor_Gear(timestep));

    Simulation.Acceleration(timestep) = ...
        Simulation.Accelerations(Simulation.Motor_Gear(timestep));

    Simulation.Motor_Speed(timestep) = ...
        Simulation.Differential_Speed ...
        * Vehicle.Motor_Gear_Ratios(Simulation.Motor_Gear(timestep));
%If there aren't any valid displacements, Hydrostatic_Drive is called
else
    Hydrostatic_Drive;
end
```

Appendix T: Interp_Reduction.m

```
%% Interp_Reducion.m
% This function finds a range of indicies around the target value
% to incrase the speed interpolation for large arrays

function [Indicies] = Interp2_Reduction(Vector, Value, Range)

    if ((Value > max(Vector)) || (Value < min(Vector)))
            Indicies = 1:(2*Range +1);
    else
        index = find(Vector >= Value,1);

        if (index <= Range)
            Indicies = 1:(2*Range +1);
        elseif (index >= (length(Vector)-Range))
            Indicies = (length(Vector)-2*Range):length(Vector);
        elseif (~isempty(index))
            Indicies = (index-Range):(index+Range);
        end
    end

end
```

Appendix U: Hydrostatic_Drive.m

```
%% Hydrostatic_Drive.m
% This file determines the motor parameters when driving in hydrostatic
% mode.
% interp2_mod is a modified version of interp2. Some error checking is
% removed to speed execution. The file is not included.


%Set mode values
Simulation.Hydrostatic(timestep) = true;
Simulation.idle(timestep) = false;

%keep the engine running to minimize cycling
Simulation.charge(timestep) = false;

Simulation.avg_diff_speed = ...
        ( ...
            Simulation.Differential_Speed ...
            + Simulation.acc*Simulation.dt ...
            / (2*pi*Vehicle.Tire_Diameter)*60*Vehicle.Differential_Ratio);

Simulation.avg_motor_speeds = ...
        Simulation.avg_diff_speed.*Vehicle.Motor_Gear_Ratios;

%Gearbox efficiency is already accounted for
Simulation.Torques = Simulation.Torque./Vehicle.Motor_Gear_Ratios;



if (isempty(Simulation.Effective_BSFCs))
    for index2 = 1:length(Vehicle.EP_Gear_Ratios)

            Simulation.Effective_BSFCs(index2,:) = ...
                    interp2_mod( ...
                        Motor.Speed_Range,...
                        Motor.Torque_Range, ...
                        Hydrostatic.Effective_BSFC(:,:,index2), ...
                        Simulation.avg_motor_speeds, ...
                        Simulation.Torques);

    end
    [Simulation.Effective_BSFCs Simulation.EP_Gears]= ...
            min(Simulation.Effective_BSFCs);
end

%Find the minimum effective BSFC
[Simulation.Effective_BSFCs Simulation.Motor_Gear(timestep)] = ...
        min(Simulation.Effective_BSFCs);

Simulation.EP_Gear(timestep) = ...
        Simulation.EP_Gears(Simulation.Motor_Gear(timestep));

Simulation.avg_motor_speed = ...
```

```
        Simulation.avg_motor_speeds(Simulation.Motor_Gear(timestep));

Simulation.Torque = Simulation.Torques(Simulation.Motor_Gear(timestep));

Simulation.Torque_Indices = [ ];
Simulation.Speed_Indices = [ ];
Simulation.Torque_Indices = Interp_reduction( ...
                                Motor.Torque_Range, ...
                                Simulation.Torque, ...
                                Simulation.Interp2_Range);
Simulation.Speed_Indices = Interp_reduction( ...
                                Motor.Speed_Range, ...
                                Simulation.avg_motor_speed, ...
                                Simulation.Interp2_Range);


Simulation.Motor_Displacement(timestep) = ...
        interp2_mod( ...
            Motor.Speed_Range(Simulation.Speed_Indices), ...
            Motor.Torque_Range(Simulation.Torque_Indices), ...
            Hydrostatic.Motor_Displacement( ...
                Simulation.Torque_Indices, ...
                Simulation.Speed_Indices, ...
                Simulation.EP_Gear(timestep)), ...
            Simulation.avg_motor_speed, ...
            Simulation.Torque);

Simulation.Motor_Mech_Efficiency(timestep) = ...
        interp2_mod( ...
            Motor.Speed_Range(Simulation.Speed_Indices), ...
            Motor.Torque_Range(Simulation.Torque_Indices), ...
            Hydrostatic.Motor_Mech_Efficiency( ...
                Simulation.Torque_Indices, ...
                Simulation.Speed_Indices, ...
                Simulation.EP_Gear(timestep)), ...
                Simulation.avg_motor_speed, ...
                Simulation.Torque);

Simulation.Motor_Torque(timestep) = Simulation.Torque;

Simulation.Acceleration(timestep) = ...
        ( ...
            Simulation.Motor_Torque(timestep)*Vehicle.Differential_Ratio ...
            *Vehicle.Gearbox_Efficiency ...
            * Vehicle.Motor_Gear_Ratios(Simulation.Motor_Gear(timestep))...
            / (Vehicle.Tire_Diameter/2) ...
            - 1/2*Environment.Air_Density*Vehicle.CdA ...
            * ( ...
                Simulation.Velocity(timestep)^2 ...
                + Simulation.Velocity(timestep) ...
                    * Simulation.acc*Simulation.dt ...
                + Simulation.acc^2*Simulation.dt^2/3 ...
              ) ...
            - ( ...
                Vehicle.RL.C*(Simulation.Velocity(timestep)*3600/5280)^2 ...
```

```
                + Vehicle.RL.C*(Simulation.Velocity(timestep)*3600/5280) ...
                    *Simulation.acc*Simulation.dt*3600/5280 ...
                + Vehicle.RL.C ...
                    *(Simulation.acc*3600/5280)^2*Simulation.dt^2/3 ...
                + Vehicle.RL.B ...
                    *(Simulation.Velocity(timestep)*3600/5280) ...
                + Vehicle.RL.B ...
                    *(Simulation.acc*3600/5280)*Simulation.dt/2 ...
                + Vehicle.RL.A ...
              ) ...
        ) ...
        / ( ...
            Vehicle.Weight/32.2 ...
            + Vehicle.Wheel_MOI/((Vehicle.Tire_Diameter/2)^2) ...
          );

%interpolate motor parameters
Simulation.Motor_Vol_Efficiency(timestep)= ...
        interp2_mod( ...
            Motor.Speed_Range(Simulation.Speed_Indices), ...
            Motor.Torque_Range(Simulation.Torque_Indices), ...
            Hydrostatic.Motor_Vol_Efficiency( ...
                Simulation.Torque_Indices, ...
                Simulation.Speed_Indices, ...
                Simulation.EP_Gear(timestep)), ...
            Simulation.avg_motor_speed, ...
            Simulation.Torque);

Simulation.Motor_Flow_Rate(timestep)= ...
        interp2_mod( ...
            Motor.Speed_Range(Simulation.Speed_Indices), ...
            Motor.Torque_Range(Simulation.Torque_Indices), ...
            Hydrostatic.Motor_Flow_Rate( ...
                Simulation.Torque_Indices, ...
                Simulation.Speed_Indices, ...
                Simulation.EP_Gear(timestep)), ...
            Simulation.avg_motor_speed, ...
            Simulation.Torque);

Simulation.Motor_Speed(timestep) = Simulation.Differential_Speed ...
        * Vehicle.Motor_Gear_Ratios(Simulation.Motor_Gear(timestep));

Simulation.Hydrostatic_Pressure(timestep) = ...
        interp2_mod( ...
            Motor.Speed_Range(Simulation.Speed_Indices), ...
            Motor.Torque_Range(Simulation.Torque_Indices), ...
            Hydrostatic.Pressure(Simulation.Torque_Indices, ...
            Simulation.Speed_Indices,Simulation.EP_Gear(timestep)), ...
            Simulation.avg_motor_speed, ...
            Simulation.Torque);
```

Appendix V: Save_Results.m

```
%% Save_Results.m
% This file saves the results of the simulation. File names are
% incremented.

files = dir('.\Data\Sim_Results*.mat');
if (~isempty(files))
    save(['.\Data\Sim_Results' num2str(length(files)+1) '.mat'], ...
            'Sim_Results')
else
    save('.\Data\Sim_Results1.mat', 'Sim_Results')
end
```